PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-006158
(43)Date of publication of application : 10.01.2003

(51)Int.Cl. G06F 15/00
G06F 13/00
// G06F 17/30

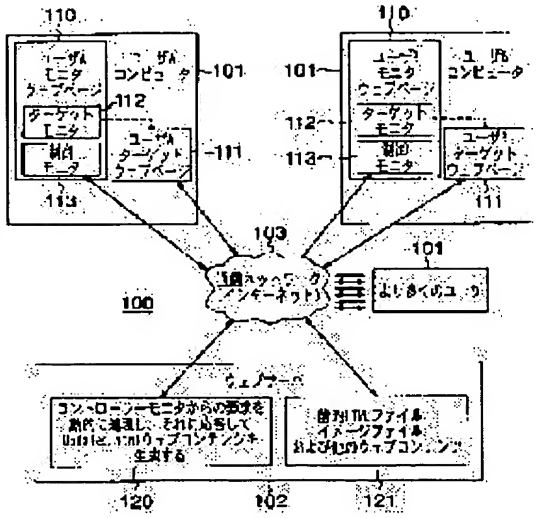
(21)Application number : 2002-079048
(22)Date of filing : 20.03.2002

(71)Applicant : MITSUBISHI ELECTRIC RESEARCH LABORATORIES INC
(72)Inventor : ESENTHER ALAN W

(30)Priority
Priority number : 2001 814190 Priority date : 21.03.2001 Priority country : US

(54) METHOD FOR COMMONLY BROWSING WEB CONTENT

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a flexible mechanism that synchronizes a plurality of web browsers, and also to enable web browser, which do not change to share an web page which do not change.
SOLUTION: A computerized system enables multiple users of the standard internet web browsers to collaborate by having significant states of their browser, such as which web page is currently being viewed, scrollbar positions, and form values, to be controlled remotely by the users of other Internet web browsers. The system uses a monitor to poll the static and dynamic state of the selected pages, and to communicate is state to a controller in middle of performing over a web server.



LEGAL STATUS

[Date of request for examination] 09.03.2005
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2003－6158
（P2003－6158A）
(43)公開日　平成15年１月10日(2003.1.10)

(51)Int.Cl.⁷		識別記号	FI		テーマコード(参考)
Ｇ０６Ｆ	15/00	３１０	Ｇ０６Ｆ 15/00	３１０Ｒ	５Ｂ０７５
	13/00	６５０	13/00	６５０Ａ	５Ｂ０８５
// Ｇ０６Ｆ	17/30	１１０	17/30	１１０Ｆ	

審査請求　未請求　請求項の数12　ＯＬ　外国語出願　（全 60 頁）

(21)出願番号　特願2002－79048(P2002－79048)

(22)出願日　平成14年３月20日(2002.3.20)

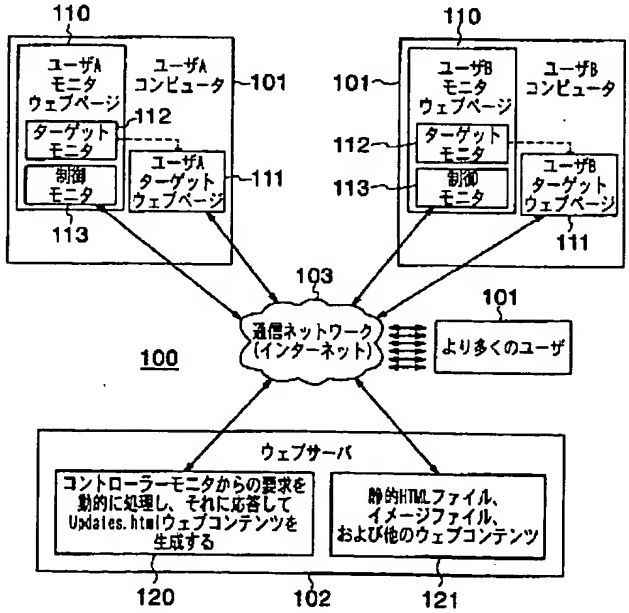
(31)優先権主張番号　０９／８１４１９０
(32)優先日　平成13年３月21日(2001.3.21)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　597067574
ミツビシ・エレクトリック・リサーチ・ラボラトリーズ・インコーポレイテッド
アメリカ合衆国、マサチューセッツ州、ケンブリッジ、ブロードウエイ　201
201 BROADWAY, CAMBRIDGE, MASSACHUSETTS 02139, U. S. A.
(74)代理人　100057874
弁理士　曾我　道照　（外６名）

最終頁に続く

(54)【発明の名称】　ウェブコンテンツを共同的に閲覧する方法

(57)【要約】　（修正有）
【課題】複数のウエブブラウザを同期する柔軟な機構を提供する。また、変更のないウエブブラウザが変更のないウエブページを共有できるようにする。
【解決手段】コンピュータ化されたシステムは、どのウェブページが現在閲覧されているか、スクロールバーの位置、およびフォーム値等のブラウザの有意な状態を、他のインターネットウェブブラウザのユーザにより遠隔的に制御させることにより、標準インターネットウェブブラウザの複数ユーザがコラボレーションできるようにする。本システムは、モニタを使用して、選択されたページの静的状態および動的状態をポーリングし、ウェブサーバ上で実行中のコントローラにその状態を連絡する。



【特許請求の範囲】
【請求項１】　複数のクライアントコンピュータおよびサーバコンピュータを含むネットワークでのウェブコンテンツを共同的に閲覧する方法において、
第１のウェブブラウザインスタンスに表示されるウェブページの静的状態および動的状態をポーリングするステップと、
前記ネットワークを介して前記状態をコントローラに送信するステップと、
前記コントローラにおいて、前記静的状態および動的状態を含む更新メッセージを作成するステップと、
第２のウェブブラウザインスタンスにおいて、前記コントローラをポーリングし、前記ネットワークを介して前記更新メッセージを受信するステップと、
前記第２のウェブブラウザインスタンスにおいて、前記更新メッセージ中の前記状態に従って前記ウェブページを表示し、前記第２のウェブブラウザインスタンスを前記第１のウェブページインスタンスに動的に同期させるステップとを備えたことを特徴とする方法。
【請求項２】　前記第１および第２のインスタンスからなる複合的なインスタンスを備えたことを特徴とする請求項１に記載の方法。
【請求項３】　前記状態をポーリングするステップは、前記第１のウェブブラウザインスタンスにおいて動作中のモニタにおいて実行され、
前記コントローラをポーリングするステップは、前記第２のウェブブラウザインスタンスにおいて実行されることを特徴とする請求項１に記載の方法。
【請求項４】　検索されたターゲットウェブページは、サーバに格納されることを特徴とする請求項１に記載の方法。
【請求項５】　前記ポーリングするステップは、ポーリング要求によって実行されることを特徴とする請求項１に記載の方法。
【請求項６】　前記第１および第２のウェブブラウザインスタンスは、前記クライアントコンピュータにおいて実行され、
サーバは、前記サーバコンピュータにおいて実行されることを特徴とする請求項１に記載の方法。
【請求項７】　前記ウェブページは、前記第１のウェブブラウザインスタンスのユーザによって選択されることを特徴とする請求項１に記載の方法。
【請求項８】　前記ウェブページおよびすべてのウェブブラウザインスタンスは、前記共同的な閲覧セッション中に変更されないままであることを特徴とする請求項１に記載の方法。
【請求項９】　前記ポーリング要求に対する応答は、隠しブラウザフレーム、隠しレイヤ、または別のブラウザウィンドウにロードされることを特徴とする請求項４に記載の方法。
【請求項１０】　前記隠しブラウザフレームはゼロ次元であることを特徴とする請求項９に記載の方法。
【請求項１１】　前記コントローラは、前記更新メッセージを動的に変換することを特徴とする請求項１に記載の方法。
【請求項１２】　前記第２のウェブブラウザインスタンスに表示される別のウェブページの静的状態および動的状態をポーリングするステップと、
前記ネットワークを介して前記状態を前記コントローラに送信するステップと、
前記コントローラにおいて、前記静的状態および動的状態を含む更新メッセージを作成するステップと、
前記第１のウェブブラウザインスタンスにおいて、前記コントローラをポーリングし、前記ネットワークを介して前記更新メッセージを受信するステップと、
前記第１のウェブブラウザインスタンスにおいて、前記更新メッセージ中の前記状態に従って他のウェブページを表示して、前記第１のウェブブラウザインスタンスを前記第２のウェブブラウザインスタンスに動的に同期させるステップとをさらに備えたことを特徴とする請求項１に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、概してコンピュータネットワークの分野に関し、特に、複数ユーザによるネットワークコンテンツに対するアクセスの同期に関する。
【０００２】
【従来の技術】インターネット閲覧アプリケーションによっては、すべてのユーザが同時に同じ「ターゲット」ウェブページを閲覧することができるように、複数のウェブブラウザを同期させることが有意なものがある。このタイプの共同閲覧は通常、電話での会話、教室での講義、またはおそらくコラボレーション（共同作業）ユーザの間でのインターネットベースの音声またはテキスト通信と併せて行われる。
【０００３】いくつかの共同閲覧システムが、従来技術において説明されている。たとえば、米国特許第６，０３５，３３２号「METHOD FOR MONITORING USER INTERACTIONS WITH WEB PAGES FROM WEB SERVER USING DATA AND COMMAND LISTS FOR MAINTAINING INFORMATION VISITED AND ISSUED BY PARTICIPANTS」、米国特許第５，９４４，７９１号「COLLABORATIVE WEB BROWSER」、米国特許第６，００９，４２９号「HTML GUIDED WEB TOUR」、米国特許第５，９９１，７９６号「TECHNIQUEFOR OBTAINING AND EXCHANGING INFORMATION ON THE WORLD WIDE WEB」、米国特許第６，１５１，６２２号「METHOD AND SYSTEM FOR PORTABLY ENABLING VIEW SYNCHRONIZATION OVER THE WORLD-WIDE WEB USING FRAME HIERARCHIES」、および米国特許第５，８０９，２４７号「METHOD

AND APPARATUS FOR GUIDED TOURINGOF INTERNET/INTRANET WEBSITES」を参照されたい。

【０００４】殆どの従来技術による共同閲覧システムでは、少なくとも１人のユーザが「マスタ」すなわち送信者と指定され、他のユーザが、「スレーブ」すなわち受信者と指定される。通常、送信者は、閲覧者により共同的に閲覧されるターゲットウェブページを選択する。システムによっては、受信者もまた送信者の役割を引き受ける能力を有するものもある。

【０００５】後述するように、既存の共同閲覧システムは、閲覧者を選択されたページに導くために使用される機構に関して大まかに分類することができる。

【０００６】１つの分類のシステムは、「プッシュ」機構を使用する。この場合、ウェブサーバは、特殊なＨＴＴＰヘッダ（Content-Type = multipart/mixed; boundary ="someString"）に頼って、サーバと受信者のクライアントコンピュータで実行されているブラウザとの間に専用ＨＴＴＰ接続を開いたままにする。このアプローチでは、送信者のブラウザが、選択された各ウェブページのＵＲＬをサーバで実行されている制御プログラムに送信する。次に、サーバのコントローラがウェブページを取り込み、おそらくそのコンテンツを処理し、専用ＨＴＴＰ接続を介してそのページを各クライアント受信者にプッシュする。

【０００７】この分類のシステムには、多数の問題がある。最も深刻な問題は、プッシュ機構が、現在入手可能なすべてのインターネットウェブブラウザ、たとえばＭｉｃｒｏｓｏｆｔ（登録商標）のＩｎｔｅｒｎｅｔ Ｅｘｐｌｏｒｅｒによってサポートされていないことである。また、将来の要求をコントローラに配向するために、ウェブページのコンテンツを、送信前に変更する必要が生じる場合がある。たとえば、ページ中のすべてのリンクを、このような要求が将来コントローラを通してプロキシされるように処理する必要がありうる。この技法は、単純なウェブページに対してしか実行することができない。複雑なページは、送信者の制御なしで新しい要求をトリガしうる多くの異なるＨＴＭＬタグおよびＪａｖａ（登録商標）スクリプトコマンドを有する可能性がある。これには、少なからぬ量の努力が必要であり、場合によっては、このようにして任意の複雑なウェブページの完全な処理に遅れを生じさせることがある。

【０００８】さらに、受信者がブラウザをある他のページに配向すると、専用ＨＴＴＰ接続が失われる。ターゲットウェブページを、受信者への送信前に動的に処理することにより、ターゲットウェブページ上のすべてのリンクをディセーブルすることが可能である。しかし、これは、受信者の役割を完全に受け身的な観察者の役割に制限する。また、リンクのディセーブル化は、受信者がある他の方法を使用して異なるウェブページをロードする場合には機能しない。より重要なことには、ページの変更により、意図される挙動が変化する可能性がある。この解決策はまた、後述するように、ターゲットウェブページがＨＴＭＬフレームを含む場合、またはキャッシングがイネーブルされる場合にも困難を呈する。

【０００９】最後に、プッシュ機構を使用する場合、受信者はいずれの種類の制御も持つことができない。ユーザは、そのページが何であれ送信者によって選択されたページに単に「プッシュ」されるだけである。

【００１０】別の分類の共同閲覧システムでは、特別なモニタが、コラボレーションを行う各コンピュータにインストールされる。モニタは、コントローラに対して専用ソケット接続を開く。この場合、送信者のモニタは、ウェブブラウザのアクティビティを追跡し、更新についての情報をコントローラに連絡し、コントローラにおいて更新を受信者のモニタに中継することができる。特定のブラウザの実施に特異的になりすぎることとは別に、ユーザがコラボレーション可能になる前に、モニタプログラムを各ユーザコンピュータにインストールする必要がある。特化されたプログラムのインストールを必要とせず、したがって、ユーザが、最初に共同閲覧する場合であっても、直ちにコラボレーションすることができる解決策を見つけることがはるかに望ましい。

【００１１】別の分類の共同閲覧システムでは、ブラウザ「プラグイン」がウェブブラウザにインストールされる。プラグインは、上記モニタと同様に挙動し、同じ欠点を有する。

【００１２】Ｍｉｃｒｏｓｏｆｔ ＮｅｔＭｅｅｔｉｎｇ（登録商標）等のアプリケーションにより具現化される別の分類の共同システムは、ネットワークにわたりユーザの画面のすべてまたは一部を本質的に共有することによってコラボレーションを促進する。これらは、より大きな帯域幅、実行可能バイナリの事前インストール、事前登録、およびユーザトレーニングが必要なため、複雑であり、プラットフォーム固有であり、また「重い」アプローチであるとみなすことができる。インストールされた実行可能バイナリをベースとする他のアプリケーションにおけるように、このような非ウェブベースのアプローチは、ソフトウェアのインストール、コンフィギュレーション、およびメンテナンスの問題をもたらす。任意のユーザは、皆が偶然同じプラットフォームを使用し、偶然インストール、登録、およびトレーニングの要件を事前に満たしており、かつユーザの間に十分な帯域幅がある場合以外は、このようなシステムを使用してコラボレーションすることができない。このような要件をいずれも持たない解決策を見つけることが望ましい。

【００１３】別の分類のシステムは、Ｊａｖａ（登録商標）アプレットや同様の技術等の特別なダウンロード可能プログラムを使用する。アプレットは各ウェブブラウザで実行され、コントローラに対する専用ソケット接続を開く。しかし、このようなシステムは、Ｊａｖａ（登

録商標）がブラウザでディセーブルされている場合には機能しない。Ｊａｖａ（登録商標）がイネーブルされている場合であっても、単純なＤＨＴＭＬウェブページのロードにかかる時間よりも、Ｊａｖａ（登録商標）環境をロードして開始する時間量のほうが依然としてはるかに多く、パフォーマンスを劣化させる。さらに、通常、この分類のシステムでは、コードをターゲットウェブページに埋め込む、すなわちアプレットをロードするための「タグ」をターゲットウェブページに埋め込む必要がある。これにより、共同閲覧はこのような特別に準備されたウェブページだけに限定される。特別なコードが、ページが取り込まれるときに動的にページに埋め込まれる場合には、ページに対する変更が、意図される挙動を変更することになる危険性がある。ターゲットウェブページの変更を必要としないよりよい解決策が望ましい。

【００１４】既存のシステムは、ターゲットウェブページがＨＴＭＬフレームを使用している場合に他の問題を有しうる。新しいウェブページが送信者のブラウザ上のフレームにロードされる場合、そのウェブページの最上位ＵＲＬは同じままである。換言すれば、ブラウザの「アドレス」または「ロケーション」バーにおけるＵＲＬは変化しない。したがって、送信者のブラウザにロードされた最上位ＵＲＬのみを監視するシステムは、送信者のブラウザ上のフレームが、受信者のブラウザ上の対応フレームに表示されているウェブページとは異なるウェブページを表示していることに気付かないことがある。送信者のブラウザのフレームにおいてＵＲＬが変化したことがある手段で検出される場合であっても、新しいＵＲＬにおけるウェブページを各受信者の対応フレームに強制的にロードさせるという問題が依然として存在する。送信者のブラウザ上のあらゆるフレームに対する変更が、受信者のブラウザに反映される解決策を見つけることが望ましい。

【００１５】たとえば、２０００年１１月２１日付けでFraenkel他に付与された、米国特許第６，１５１，６２２号「Method and system for portably enabling view synchronization over the world-wide web using frame hierarchies」は、選択されたフレームの階層の記述を生成することにより、クライアントウェブブラウザを選択されたフレームに対して静的に同期させることを記載している。この階層は、ネットワーク環境を介して送信され、クライアントブラウザのターゲットフレームに複製されている。

【００１６】この種類の共同閲覧には、多数の問題がある。まず、配布される獲得フレームの階層は、静的フレーム情報、具体的にはフレームの深さおよび名称、およびフレームコンテンツを表すＵＲＬしか含まない。選択されたフレームの動的状態については、何もわからない、すなわち何も獲得されない。たとえば、ユーザの１人が水平または垂直にスクロールする、または「Page Down」キーを押下すると、参加者のビューはもはや同期しなくなる。各ブラウザで閲覧中のウェブページの動的状態も同期される共同ウェブ閲覧解決策を提供することが望ましい。その場合には、たとえば、任意のユーザがフォームフィールドをスクロールする、または何かをフォームフィールドにタイプすると、その他のユーザのブラウザが、このような動的状態の変更を反映するように更新される。

【００１７】さらに、Fraenkel他によって提供される解決策では、共同閲覧するウェブページをフレームにロードする必要がある。このアプローチには多数の欠点がある。ウェブページによっては、ブラウザの最上位文書ではなくフレームにロードされる場合、適切に挙動しないものがある。フレーム自体を含むウェブページは、ブラウザの機能を停止させる傾向がより強い。フレームに強制的にロードされるウェブページは、ブックマークに追加することもできない。最後に、ウェブページがフレームに強制的にロードされると、そのウェブページのＵＲＬが、ウェブブラウザの「アドレス」または「ロケーション」フィールドで見ることができなくなる。これは、最上位ウェブページのＵＲＬのみがこのフィールドに示されるためである。ウェブページを共同的に閲覧されていない場合と全く同じように「自然な環境」で閲覧されることができるように、共同閲覧されるウェブページをウェブブラウザの最上位文書としてロードすることができる共同ウェブ閲覧解決策を提供することが望ましい。

【００１８】最後に、Fraenkel他によって提供される解決策は、各ブラウザにダウンロードされるコラボレーションスクリプトにインテリジェンスを配置する。別の解決策は、サーバがコラボレーションセッションに価値を追加することができるように、このインテリジェンスをクライアントではなくサーバに配置する。たとえば、サーバ上のプログラムは、コラボレーション情報の動的状態または静的状態を、その情報がクライアントブラウザに配布される前に変換することができる。一般に、サーバプログラムは補助情報源に対して素早いアクセスを有することから、インテリジェンスをサーバプログラムに保持することが有利であろう。このような情報源としては、バックエンドデータベース、クライアントによって使用されるブラウザの銘柄およびバージョンについての情報、各クライアントの画面解像度等が挙げられる。柔軟であり、共同ウェブ閲覧セッションへの価値追加のサポートへの拡張が容易な共同ウェブ閲覧解決策を提供することが望ましい。

【００１９】従来技術によるシステムは、新しいターゲットウェブページに対する送信者のＨＴＴＰ要求がホストウェブサーバに到来することに依存する場合に他の問題を有しうる。これは、ホストへのＨＴＴＰ要求の到来が、送信者による新しいターゲットウェブページ要求を処理しなければならないときを決定するベースとして用い

られる場合でありうる。問題は、そのウェブページがホストサーバ、またはある中間プロキシサーバにキャッシュされる場合があることである。送信者のブラウザによるＨＴＴＰ要求がキャッシュヒットによって満足される場合、その要求が受信者のウェブブラウザに反映されないことがある。

【００２０】

【発明が解決しようとする課題】したがって、動的状態情報を含み、任意の数のＨＴＭＬフレーム、リンク、またはＪａｖａ（登録商標）スクリプトを含む、既存の任意のターゲットウェブページについて動作する共同ウェブ閲覧システムを提供することが好ましい。また、ターゲットウェブページを自然に、かつ意図される挙動を変えうる変更なく、閲覧できることも望ましい。アクセス可能性および使用し易さのため、システムが、Ｊａｖａ（登録商標）アプレットまたはブラウザプラグイン等、特別な実行可能バイナリまたはプログラムをいずれも必要としないことも望ましい。

【００２１】

【課題を解決するための手段】本発明は、複数のウェブブラウザを同期する柔軟な機構を提供する。本発明は、変更のないウェブブラウザが変更のないウェブページを共有できるようにする。このコラボレーションは、部分的に、任意のコラボレーション中のユーザが、「プル」またはポーリング機構を使用して他ユーザのブラウザの重要な態様を遠隔制御することができることから可能になる。

【００２２】「送信者」ブラウザが、「受信者」ブラウザによって閲覧されるウェブページを選択する。受信者ブラウザは、１つまたは複数の送信者ブラウザによって「制御」可能である。任意のユーザブラウザは、送信者または受信者、あるいはこれら双方であることが可能である。本発明は、送信者ブラウザの状態の動的変化、たとえば送信者のブラウザウィンドウのサイズ変更、新しいターゲットページの選択、送信者のブラウザウィンドウまたはフレームのスクロール、ポインタ位置の追跡、およびテキスト入力、をポーリングすることによって動作する。動的状態のすべての変化が、受信者のブラウザに反映される。

【００２３】本発明は、３つの主な構成要素を備える。すなわち、ホストウェブサーバ上で実行されるコントローラ、送信者としてであれ、受信者としてであれ、それともこれら双方としてであれ、共同ウェブ閲覧システムに参加している各ユーザのモニタおよびターゲットである。

【００２４】コントローラは、コラボレーション中の各種ブラウザの動的状態の更新を調整する。コントローラの実施に使用される言語に対する制約はない。たとえば、コントローラは、任意選択的に、ＰＥＲＬスクリプトとして、またはＪａｖａ（登録商標）サーブレットとして実施されて、プラットフォーム独立性を提供することができる。

【００２５】モニタは、ＨＴＭＬフレームと、構成コントロールパネルと、Ｊａｖａ（登録商標）スクリプトプログラムとを含むＨＴＭＬウェブページを含む。モニタは、それ自体のブラウザのインスタンスにおいて、またはターゲットのフレームの１つにおいて実行することができる。モニタは、ターゲットブラウザウィンドウの名称を知っているため、Ｊａｖａ（登録商標）スクリプトプログラムは、ターゲットウィンドウの動的状態のあらゆる変更を検出することができる。送信者のモニタは、送信者のターゲットをポーリングして、コントローラに送信する動的状態の更新を検出する。たとえば、送信者がターゲットウェブページにおいて垂直スクロールバーを下にスクロールすると、送信者のモニタがこのスクロールを検出し、スクロール中状態をコントローラに送信する。各受信者のモニタは、周期的にコントローラをポーリングして、受信者のターゲットに反映すべき動的状態更新を検出する。

【００２６】ターゲットは、共同的に閲覧されるターゲットウェブページを含むブラウザのインスタンス、またはそのフレームの１つである。本発明によれば、ターゲットは、任意の標準ブラウザの普通のインスタンスであることができるため、通常のウェブ閲覧に使用することが可能である。したがって、本発明は、予め存在する任意のウェブページを、変更のないブラウザによって共同的に閲覧できるようにする。

【００２７】上述したように、１つのブラウザインスタンスまたはフレームで実行される送信者のモニタは、ターゲットブラウザインスタンスまたはフレームのインスタンスにロードされるターゲットウェブページの状態を徹底的に検査する。モニタは、１秒に１回等の構成可能な時間間隔で周期的にこのポーリングを行う。この検査は、現在ロードされているターゲットウェブページの構造を決定し、コントローラに送信すべき有意の値を確認する。単純なウェブページの場合、この情報は、ページにロードされた最上位ＵＲＬと、水平および垂直のスクロールバーの位置とを最低限含む。

【００２８】フレームを使用するウェブページの場合、情報は、各フレームに使用されるＵＲＬ、および各フレームについてのスクロールバー情報も含む。他の状態情報としては、ウェブページ上の任意のフォームにタイプされた値、現在選択されているテキスト、およびターゲットウェブページに関連する情報を挙げることができる。さらに、ウェブコンテンツがすべてのユーザのウェブブラウザで全く同じに表示されるように、ターゲットブラウザウィンドウの次元を同期することができる。このような状態値のいずれかに対する変更は、ポーリング要求、たとえばＨＴＴＰ要求、ＨＴＴＰＳ要求、または他の通信プロトコルでの同様の通知の形態でコントロー

ラに送信される。

【００２９】Ｊａｖａ（登録商標）スクリプトおよび標準ウェブブラウザに構築されたセキュリティ対策により、第１のブラウザインスタンス（またはフレーム）において実行されるＪａｖａ（登録商標）スクリプトプログラムは、第１および第２双方のインスタンスに同じ発信源からのウェブページがロードされる場合に、第２のブラウザインスタンス（またはフレーム）における詳細な情報にアクセスすることだけができる場合がある。この場合、ウェブページ、イメージ、および本発明の実施に使用されるプログラムは、共同的に閲覧されるウェブページを格納するウェブサーバと同じウェブサーバに格納されるべきである。

【００３０】送信者が１人の場合、送信者のターゲットの状態は、すべての受信者のターゲットに反映される。送信者が複数人の場合、受信者は、任意の送信者の状態と同期される。特に、受信者は、最も最近更新を行った送信者の状態に同期される。各送信者は、すべての参加ブラウザが互いに正しく同期されるように、受信者として挙動することも可能である。ユーザは、送信者でもなく受信者でもないように決めることもできる。このような場合は、たとえば、ユーザが、コラボレーションセッションをしばらく去って他のどこかを閲覧してから、後に共同閲覧セッションに戻って再度参加する場合でありうる。

【００３１】送信者のモニタが、スクロールバーの位置等のある状態が変化したことを示す更新ポーリング要求をコントローラに送信すると、コントローラは、新しい状態についての詳細を含む更新メッセージを動的に生成する。この更新メッセージは、周期的なポーリング要求に応答して、受信者のモニタに戻される。状態の変化は各受信者のブラウザインスタンス（またはフレーム）に反映されるため、各受信者のブラウザが他のすべてのブラウザと同期することが保証される。送信者またはコントローラの場合であれ、受信者の場合であれ、ターゲットにおける状態の変化についてモニタポーリングを行う頻度は、コントロールパネルを介して構成可能である。

【００３２】本発明の方法は、動的ＨＴＭＬ技術を用い、コラボレーションが開始可能になる前に、Ｊａｖａ（登録商標）アプレット、ブラウザプラグイン、または他の実行可能プログラム等の特別なプログラムをいずれもユーザコンピュータにインストールする必要がない。したがって、インターネット上の任意のユーザが、初期ウェブページを単に訪れるだけで、直ちにインターネット上の他のユーザとコラボレーションすることができる。ユーザは、従来技術のＪａｖａ（登録商標）アプレットに関連する遅延等のいかなる初期遅延も経験しない。ユーザは、Ｊａｖａ（登録商標）および同様の「重い」技術をサポートしない、限られた資源を有する装置を使用している場合であっても参加することが可能である。

【００３３】本発明による共同閲覧システムは、プラットフォームから独立し、ホストウェブサーバで実行されるコントローラは、ＰＥＲＬまたはＪａｖａ（登録商標）等のプラットフォームから独立した言語で書くことができる。

【００３４】本発明は、既知の共同ウェブ閲覧システムよりも高い柔軟性を可能にする。受信者は、定期的にどこか他を閲覧してから、後にコラボレーションセッションに再び参加することができる。１つの送信者ブラウザにすべての対話を制御させるのではなく、各ユーザは、任意選択的に送信者となることができるため、他の受信者によって共同的に閲覧されるウェブページを制御することができる。本発明はウェブベースのアプリケーションであるため、ユーザコンピュータのインストール、構成、またはメンテナンスを一切行う必要なく、新しい機能に対するサポートをシステムに追加することが可能である。任意のこのような追加は、ホストウェブサーバ上のファイルに対して行う必要があるだけである。

【００３５】より具体的に、本発明は、複数のクライアントコンピュータ、およびサーバコンピュータを含むネットワークにおいてウェブコンテンツを共同的に閲覧する方法を提供する。第１の送信者ウェブブラウザインスタンスに表示されるウェブページの静的状態および動的状態が、ポーリングされる。この状態は、ネットワークを介して、サーバコンピュータで実行されているコントローラに送信される。コントローラは、更新メッセージの受信に応答して、状態を含む更新メッセージを生成する。次に、コントローラは、第２の受信者ウェブブラウザインスタンスによってポーリングされて更新メッセージを受信し、次に、受信者のウェブブラウザが送信者のウェブブラウザと動的に同期するように、更新メッセージ中の状態に従って受信者ブラウザが表示可能であり、またそのようにウェブページを表示可能である。

【００３６】

【発明の実施の形態】図１は、本発明による共同ウェブ閲覧システム１００を示す。システム１００は、ネットワーク１０３によって互いに接続された、１つまたは複数のクライアントコンピュータ（クライアント）システム１０１およびサーバ（ホスト）コンピュータシステム１０２を含む。

【００３７】クライアント１０１は、パーソナルコンピュータ、ワークステーション、ラップトップ、個人情報端末（ＰＤＡ）、セルラ電話等の無線計算装置、およびソフトウェアプログラムを実行する同様のものであることができる。オペレーティングシステムソフトウェアは、Ｗｉｎｄｏｗｓ（登録商標）、ＬＩＮＵＸ、ＵＮＩＸ（登録商標）、ＮＴ等であることができ、アプリケーションソフトウェアは、ウェブブラウザ等のインターネットアプリケーションを含むことができる。サーバは、

アパッチシステム等のサーバソフトウェアを実行可能である。通常、クライアントおよびサーバは、入力装置、出力装置、および格納サブシステムを含む。ネットワークは、ワールドワイドウェブ（WWWまたはウェブ）であることができ、クライアントとサーバの間のリンクは有線または無線であることができる。ネットワークは、中間プロキシサーバおよびルータも備えることが可能である。

【００３８】システム１００の３つの主な構成要素としては、サーバにおけるコントローラ１２０、各クライアント１０１のモニタ１１０およびターゲット１１１が含まれる。モニタ１１０は、ターゲットモニタ１１２およびコントローラモニタ１１３を含みうる。クライアントがマルチユーザシステムである場合、クライアントは、各ユーザについてモニタのインスタンスおよびターゲットを含むことができる。

【００３９】コントローラ１２０は、サーバ１０２上で実行されるアプリケーションプログラムである。サーバは、システム１００のユーザによって共同的に閲覧される、ウェブページ、イメージファイル、ビデオファイル等のウェブコンテンツ１２１を格納することができる。コンテンツは、ネットワークのクライアント、ローカルサーバ、または中間サーバに格納され、キャッシュされてもよい。

【００４０】システム１００のユーザは、「マスタ」すなわち送信者として、「スレーブ」すなわち受信者として、これら双方として、またはどちらでもないものとして動的にイネーブル化可能である。送信者によって選択され閲覧されるコンテンツ１２１は、すべての参加受信者のターゲット１１１に反映される。ターゲットモニタ１１２は送信者に対してイネーブルされ、コントローラモニタ１１３は受信者に対してイネーブルされる。ターゲットは、各種ウェブコンテンツ１２１をロード可能な標準の変更されていないウェブブラウザのインスタンスである。

【００４１】ターゲットモニタ１１２は、Ｊａｖａ（登録商標）スクリプトプログラムを使用して、送信者のターゲット１１１の動的状態を周期的にポーリングし、動的状態のあらゆる変更をコントローラ１２０への更新として送信する必要があるかどうかを確認する。任意の送信者は、共同的に閲覧されるウェブコンテンツ１２１をその送信者のターゲットにロードすることができる。コントローラモニタ１１３は、ポーリング要求を用いてコントローラ１２０と連絡する。コントローラモニタは、送信者によって実行されたアクションに応答して生成される更新メッセージを取り込む。本発明は、モニタおよびターゲット構成要素が別個のウェブブラウザインスタンスに存在する必要がないことに留意する。本発明ではまた、このような構成要素が単一ブラウザインスタンスの異なるフレームにおいて実行してもよいが、好ましい実施形態では、２つの別個のブラウザインスタンスがある。

【００４２】図２は、本発明の好ましい実施のクライアント側２００を示し、モニタおよびターゲットが、同じウェブページの別個のフレームにあるのではなく別個のブラウザインスタンスに存在する。モニタを含むウェブブラウザインスタンスは、最初に、共同ウェブブラウザコントロールパネル２１０を表示する。このコントロールパネルは、共同閲覧セッションを構成するために使用される。ターゲットブラウザインスタンス２２０は、受信者によって共同的に閲覧されるウェブページを閲覧するために、送信者によって使用される。

【００４３】図３に示すように、モニタウェブページ３００は、３つのフレーム３０１～３０３を含む。第１のフレーム３０１は可視である。その他の２つのフレーム３０２、３０３は、ユーザにとって不可視の「隠し」ＨＴＭＬフレームである。効果的なことに、隠しフレームはゼロ次元である。すなわち、隠しフレームは高さも幅も持たない。

【００４４】モニタウェブページ３００における第１のフレーム３０１は、コントロールパネル２１０である。コントロールパネルは、共同閲覧セッションの特定の態様を構成するために使用される。ユーザは、コントロールパネルを使用して、ターゲットブラウザウィンドウ２２０の次元を特定（３１１）することができる。ユーザはまた、後述するように、送信者機能３１２をイネーブルし、受信者機能３１３をイネーブルし、また分散ポインタ（distributed pointer）３１４をイネーブルすることもできる。ユーザはまた、コントロールパネルを使用して、ユーザが送信者としてイネーブルされる場合にはターゲットウィンドウの動的状態の変化３１５を、またユーザが受信者としてイネーブルされる場合にはコントローラからの変化３１６をポーリングする頻度を特定することも可能である。第１のフレーム３０１は、ターゲットモニタ１１２およびコントローラモニタ１１３を実施するＪａｖａ（登録商標）スクリプトプログラムも含む。

【００４５】コントローラモニタは、ポーリング要求を行うことによりコントローラを周期的にポーリングする。好ましい実施形態において、要求はＨＴＴＰまたはＨＴＴＰＳプロトコルに準拠する。この要求に対する応答は、ファイル名「Updates.html」で参照されるウェブページである。Updates.htmlページは、第１の隠しフレーム３０２にロードされる。Updates.htmlファイルのコンテンツは、任意の変更が任意の送信者ユーザのターゲットウェブページの動的状態に行われたかどうかを示す。新しいウェブページが送信者のターゲットブラウザウィンドウにロードされたときなど、いくつかの変更が行われていた場合、それら変更がローカルターゲットウィンドウに適用される。コントローラモニタにおけるＪ

ａｖａ（登録商標）スクリプトプログラムは、受信者に対してのみイネーブルされる。

【００４６】ターゲットモニタ１１２は、ターゲットウィンドウ２２０に現在ロードされているウェブページの動的状態を周期的にポーリングする。ターゲットウェブページの状態に対するあらゆる変更は、コントローラ１２０に送信され、コントローラ１２０が、その変更をすべての受信者に提供する。好ましい実施形態では、このような変更は、ＨＴＴＰ要求におけるパラメータとしてコントローラに送信される。この要求に対するＨＴＴＰ応答は、第２の隠しフレーム３０３にロードされる。コントローラからのＨＴＴＰ応答は、デバッグに使用される。ターゲットモニタを実施するＪａｖａ（登録商標）スクリプトプログラムは、ローカルユーザが送信者としてイネーブルされる場合にのみイネーブルされる。

【００４７】図４は、ターゲットモニタ１１２がコンパクトな形態４０１および拡張した形態４０２で動的状態更新をコントローラ１２０に送信するときに、ターゲットモニタ１１２によって行われるポーリング要求の例を示す。この例では、HTTP GET方法が使用される。代替として、おそらく多量のデータが更新にある場合には、HTTP POST方法を使用してもよい。ターゲットモニタは、GETまたはPOST方法が特定の更新セットに最も適切であるかどうかを動的に決定することができる。

【００４８】図４に示す例では、ＨＴＴＰパラメータの名称および値は、簡潔にするために単純なテキスト置換を介して符号化される。たとえば、“frames[”という形態のストリングは、“Z[”で置換される。ターゲットウィンドウにおける各フレームについての情報は、“fn.d”という形態の照会パラメータ名を使用して符号化されることに留意する。ただし、“n”は特定のフレームを示す番号であり、“d”はそのフレームの特定の属性を示す。たとえば、“f1.S”は、“f1”で指定されるフレームにロードされるウェブページのＵＲＬを示す。このようなフレーム指定の符号化は照会パラメータに含まれてもよく、たとえば“fn=P.Z[1]-f2”は、ネストされたフレームであるframes[1].frames[0]の属性が、“f2”で始まる名称を有する照会パラメータに含まれることを示す。パラメータの名称および値を符号化する他の方法を使用してもよく、また使用される方法は、本発明が使用されている環境および通信機構に最もよく合うように、たとえばＳＯＡＰプロトコルが使用される場合には、ＸＭＬフォーマットを使用してデータを送信できるように、適応可能なことを認識されたい。

【００４９】図５は、Updates.htmlファイル５００の例を示す。これは、送信者上のターゲットモニタからの更新に応答して、コントローラ１２０によって生成されるファイルである。このファイルは、各受信者側でコントローラモニタによって周期的に取り込まれる。Updates.htmlは、モニタウェブページの第２のフレーム３０２にロードされることに留意する。擬似コード中の“onLoad”ハンドラに留意する。このハンドラは、このウェブページのブラウザへのロードが終了したときに実行される。これは、更新における動的状態属性値を受信者のターゲットウェブブラウザにおけるウェブページにおける動的状態属性値と明示的に比較する。いずれかの属性値がマッチしない場合、その値が、受信者のターゲットブラウザにおいて、Updates.html中のＪａｖａ（登録商標）スクリプトコードによって更新される。図５では、「スレーブ」という語が「受信者」を示すことに留意する。

【００５０】受信者のターゲットウェブページ２２０の更新に使用されるUpdates.htmlファイル５００における制御フローは、当業者には明白なはずである。シーケンス番号を用いて、このUpdates.htmlファイルにおいて反映された特定の変更セットがすでに処理されているか否かを確認する。任意のサブフレームにおける属性も再帰的に更新されることに留意する。ブラウザに固有のＪａｖａ（登録商標）スクリプトコマンドは、受信者のターゲットウェブページ２２０の更新に必要なときに使用することができる。これは、必要であれば、コントローラおよびモニタが、標準的なＪａｖａ（登録商標）スクリプトドキュメントオブジェクトモデル方法および属性を使用して、使用中のウェブブラウザについてのプラットフォームおよびバージョン情報を検出することが可能なためである。図５におけるＪａｖａ（登録商標）スクリプトのフォーマットおよび使用法は、受信者に送信するために更新情報を伝達する多くの潜在的な方法の１つにすぎないことが認められる。このフォーマットは、動作を明瞭に簡潔にするために好ましい実施形態において使用される。

【００５１】図６は、送信者からの更新を処理するためにコントローラ１２０によって用いられる方法６００を示す。本方法は、ポーリング要求を待つ（６１０）。有効な要求が受信される（６１５）と、ターゲットブラウザの動的状態が抽出され（６２０）、新しいUpdates.htmlファイル５００が生成される（６３０）。

【００５２】ホストウェブサーバ１０２上で実行されるコントローラプログラム１２０は、モニタからの有効な要求を待つ（６１０）ように設計される。このような要求を検出する（６１５）と、コントローラが（通常は最小レベルの機能性のために）、ターゲットブラウザのウィンドウ幅、ウィンドウ高さ、ＵＲＬ、およびスクロールバー位置をＨＴＴＰパラメータから抽出する。これは、ターゲットウェブページにおいて使用されるフレームの任意のＵＲＬおよびスクロールバー位置（もしあれば）を含む。受信者に伝達すべき追加状態処理情報についての情報を含みうる追加パラメータを復号化することもできる。たとえば、このような追加パラメータは、後述するＨＴＭＬフォームの記入方法、どのテキストが選

択されているか、および任意の分散ポインタの存在および場所についての情報（これらのうち任意のものは、ターゲットウェブページにおける任意のおそらくネストされたフレームにおいて使用可能でありうる）を含みうる。

【００５３】送信者がターゲットウェブページに対して行った更新についての情報を抽出した後、コントローラが、ホストウェブサーバ１０２上で新しいUpdates.htmlファイル５００を生成する（６３０）。図５に示すように、このUpdates.htmlファイルはＪａｖａ（登録商標）スクリプトonLoadイベントハンドラを含み、このＪａｖａ（登録商標）スクリプトonLoadインベントハンドラは、Updates.htmlファイルの受信者の隠しフレーム３０２へのロードが終了したときに実行される。このonLoadハンドラは、誤り修正を行う（たとえば、受信者のターゲットウィンドウがまだ開いており、アクセス可能であるかどうかを調べる）コード、更新シーケンス番号をチェックしてこの特定のUpdates.htmlファイルがすでに処理されているかどうかを調べるコード、さらに、任意の新しい更新を受信者のターゲットウェブページに適用するコード、そして最後に、この更新セットが適用されたことを示すように受信者のモニタの更新シーケンス番号を更新するコードを含む。

【００５４】図７ａ、図７ｂ、図７ｃは、ユーザのコントロールパネルを動作するためにモニタによって使用される方法７１０～７１９、コントローラモニタの実施に使用される方法７２０～７２３、およびターゲットモニタの実施に使用される方法７３０～７４１を示す。

【００５５】図７ａは、コントロールパネルを介してユーザによって行われる構成変更に応答して、モニタによって使用される方法を示す。図７ａに示すように、まず、コントロールパネル３０１および２つの隠しフレーム３０２～３０３を含むモニタウェブページが、モニタウィンドウまたはフレーム２１０にロードされる（７１０）。モニタコントロールパネル２１０におけるフォーム要素に関連する標準的なイベントハンドラは、ユーザによって行われる構成変更に応答するために使用される。これは、図７ａのブロック７１１に示されており、ここで方法は、ユーザがコントロールパネルにおいていくらか状態を変更するのを待つ。ユーザが、「マスタ」（送信者）のチェックボックスをチェックする（７１２）またはチェックを外す（７１３）場合、checkForChangesByMaster()におけるTargetMonitorルーチン７３０へのスケジューリングされる呼び出し（７１６）、またはスケジューリングされた既存の呼び出しのいずれかである呼び出しはキャンセルされる（７１７）。同様に、ユーザが、「スレーブ」（受信者）チェックボックスをチェックする（７１４）、またはチェックを外す（７１５）場合、checkForUpdatesToSlave()におけるControllerMonitorルーチン７２０へのスケジューリングされる呼び出し（７１８）か、あるいは既存のスケジューリングされた呼び出しのいずれかである呼び出しはキャンセルされる（７１９）。さらに、ユーザがスレーブチェックボックスをチェックする場合（７１４）、このチェックボックスが再びチェックされるときに、受信者が適宜再同期されるように、ControllerMonitorにおける“last_updates_sequence_number”カウンタがリセットされる（７１８）。リセットされない場合には、ControllerMonitorが、この受信者がすでに最新のものであると考えてしまうことがある。

【００５６】図７ｂは、送信者からの新しい更新が検出され適用されるように、コントローラを定期的に監視するために、受信者におけるコントローラモニタ１１３によって用いられる方法を示す。換言すれば、checkForUpdatesToSlave()コントローラモニタルーチンが、受信者（スレーブ）を更新するために用いられる。モニタが受信者として構成される（スレーブチェックボックス３１３がチェックされる）場合、checkForUpdatesToSlave()コントローラモニタルーチンが、定期的に呼び出される（７２０）。

【００５７】このルーチンが呼び出される頻度は、モニタコントロールパネル３１６において構成することができる。この方法は、ホストウェブサーバ１０２上のコントローラ１２０によって生成された最も近いUpdates.htmlファイル５００を用いて、単にモニタウェブページ３００における第１の隠しフレーム３０２をリロードする（７２１）。Updates.htmlファイル中のonLoadハンドラは、上述したように、必要な更新をいずれも適用する。コントローラモニタは、任意選択的に、任意のさらなる特別動作を実行して（７２２）、受信者のターゲットウェブページのコンテンツまたは状態をさらに処理することができる。このような動作は、付加価値を共同ウェブ閲覧セッションに加えるために使用することが可能である。このような動作の態様は、たとえば、更新メッセージにより、送信者または受信者のモニタコントロールパネルにおける設定により、またはユーザアクションにより制御可能である。最後に、コントローラモニタcheckForUpdatesToSlave()ルーチンが、次の更新セットも適用されるように、コントロールパネル３１６において特定される間隔後に、それ自体に対する別の呼び出しを再びスケジューリングする（７２３）。

【００５８】図７ｃは、あらゆる変更がコントローラに送信されるように、送信者のターゲットウェブページを定期的に監視するために、送信者におけるターゲットモニタ１１２によって使用される方法を示す。換言すれば、checkForChangesByMaster()ターゲットモニタルーチンは、この送信者（マスタ）によって行われるあらゆる変更をチェックし、変更があれば、受信者に中継するためにその変更をコントローラに送信するために使用される。モニタが送信者として構成される（マスタチェッ

クボックス３１２がチェックされる）場合、checkForChangesByMaster()ターゲットモニタルーチンは、定期的に呼び出される（７３０）。

【００５９】このルーチンが呼び出される頻度は、モニタコントロールパネル３１５において構成することができる。この方法はまず、ウィンドウ次元およびターゲットブラウザウィンドウにおける最上位文書の状態を最低限含む、ターゲットウィンドウの基本（静的および動的）状態を読み出す（７３１）。最上位文書の状態は、最低限、最上位ウィンドウにロードされたＵＲＬ、および水平および垂直スクロールバーの位置を含みうる。次に、ターゲットの基本的な状態が、chackForChangesByMaster()に対する前の呼び出し中に、ターゲットモニタに保存された基本状態と比較する（７３２）。

【００６０】この比較の結果を用いる前に、ターゲットモニタルーチンは、まず、任意選択的に、任意のさらなる特別動作を行って、ターゲットウェブページのコンテンツまたは状態をさらに処理することができる（７３３）。このような動作は、付加価値を共同ウェブ閲覧セッションに加えるために使用することが可能である。このような動作の態様は、たとえば、モニタコントロールパネルにおける設定により、またはユーザアクションにより制御可能である。例として、後述する分散ポインタ等の高度な機能をサポートするために、ターゲットウェブウィンドウに検出された新しい最上位ＵＲＬがある場合、このポイントにおいて、ターゲットモニタルーチンが、ターゲットウィンドウにカスタムイベントハンドラを再帰的に設定して、このような分散ポインタの存在または移動を検出することができる。

【００６１】引き続き図７において、ターゲットモニタcheckForChangesByMaster()ルーチンはチェックを行い、ターゲットウェブページがフレームを使用するかどうかを調べる（７３４）。使用していない場合、チェックを行いターゲットの基本状態が変更されたかどうかを調べる（これは、先に確認された（７３２））。ターゲット基本状態が変更されておらず、またフレームがない場合、コントローラに送信する更新がないため、本方法は、コントロールパネル３１５において特定された間隔後に、それ自体に対する別の呼び出しを単にスケジューリングするだけである（７４１）。ターゲットがフレームを有さない（７３５）が、その基本（最上位）状態が変更した場合、本方法は、モニタウェブページにおける第２の隠しフレーム３０３（第３の実際ファイル）をリロードし、すべての変更をコントローラ７４０に送信する。ここでの概念は、第２の隠しフレームをリロードするためのＨＴＴＰ要求が、コントローラに配向され、パラメータとして更新情報を含むことである。コントローラは、更新パラメータを復号化し、それに応答して、第２の隠しフレーム３０３にロードされるいくつかのウェブページ文書を送信する。この応答文書は、送信状態（たとえば、「更新の受信に成功しました」）およびデバッグから切り離れては使用されない。

【００６２】引き続き図７ｃにおいて、ターゲットモニタがフレームを使用する場合（７３４）、checkForChangesByMaster()ルーチンは、再帰的に、ターゲットウェブページ７３６におけるすべての（おそらくネストされた）フレームの静的状態および動的状態を収集し、保存する。ターゲットの基本状態が変更した場合（７３７）（先（７３２）に確認されたように）、第２の隠しフレームがリロードされ、上述したようにこれらの変更をコントローラに送信する（７４０）。ターゲットがフレームを使用し、かつターゲットの基本状態が変更された場合、本方法は、フレームにおいて変更を再帰的にわざわざ探すことはしないことに留意する。これは、ターゲットの基本最上位状態の変更をまず処理しなければならないためである。例として、最上位ＵＲＬが変更された場合、受信者は、新しい最上位ＵＲＬによって特定されるウェブページにある任意のフレームの変更が可能になる前に、新しい最上位ＵＲＬをロードする必要がある。

【００６３】ターゲットがフレームを使用するが、checkForChangesByMaster()が最後に呼び出されたときから、ターゲットの基本状態が変化していない場合（７３７）、本方法は、再帰的に古いフレーム状態と新しいフレーム状態とを比較し（７３８）、任意選択的に、任意のさらなる特別動作を実行する（７３８）（最上位文書について上述したように（７３３））。次に、引き続きターゲットがフレームを有し、かつターゲットの基本状態が変化していない場合、本方法はチェックを行い、任意の（おそらくネストされた）フレームの状態が変更されたかどうかを調べる（７３９）。変更されている場合、第２の隠しフレーム３０３がリロードされ、上述したように、これらの変更がコントローラに送信される（７４０）。

【００６４】任意のイベントにおいて、checkForChangesByMaster()に対する任意の呼び出しの最後のステップ７３０は、コントロールパネル３１５において特定された間隔後に、chackForChangesByMaster()に対する別の呼び出しをスケジューリングすることである（７４１）。

【００６５】図８は、ユーザが、送信者および受信者として（８１０）、送信者のみとして（８２０）、受信者のみとして（８３０）、および送信者でもなく受信者でもなく（８４０）動作する場合の典型的なユーザセッションについての動作フローの例を示す。この図において、「スレーブ」という語は受信者を指し、「マスタ」という語は送信者を指す。この図は、決して本発明によって網羅される動作シーケンスをすべて示すものではないが、どのように動作するように意図されるかについての概念を当業者に与えるに十分な情報を伝える。

【００６６】ユーザが共同ウェブ閲覧セッションに参加

することができる典型的な方法を、図８に示す。このユーザは、この共同ウェブ閲覧セッションにすでに参加している他のユーザと区別するために、以下任意に「ユーザ５」と呼ぶ。ユーザ５はまず、「submit（送信）」または「Collaborate（コラボレーション）」ボタンをクリックする前に、ユーザ５が任意選択的にユーザ名およびパスワードを入力する特別な「ようこそ（Welcome）」ウェブページを取り込む（８０１）。この発明は、パスワードで保護されたページを必要としないが、所望であれば、標準的なパスワード保護機構を使用してパスワードで保護されたページを使用することができる。好ましい実施形態では、モニタウェブページおよびユーザのターゲットウェブページが、別個のウェブブラウザインスタンスで実行される。これが当てはまるものと仮定し、「送信」／「コラボレーション」ボタンをクリックする（８０１）と、２つの新しいブラウザインスタンスが、ユーザ５の画面上に開かれる（８０２）。第１のインスタンスはモニタコントロールパネル２１０を含み、その他はターゲットウェブページ２２０を含む。この状況の場合、最初に「マスタ」（送信者）チェックボックス３１２および「スレーブ」（受信者）チェックボックス３１３の双方が、コントロールパネルにおいてチェックされており、システムは、スレーブチェックボックスが最初にチェックされたかのように挙動するものとする。これは、この時点で、ユーザ５のターゲットブラウザウィンドウに、現在共同的に閲覧されているウェブページがロードされることを暗に示す。（システムが最初に、マスタチェックボックスが最初にチェックされたかのように挙動する場合には、セッションに現在存在するあらゆるスレーブ／受信者のターゲットブラウザに、ユーザ５のターゲットブラウザウィンドウに最初にロードされるページと同じページが直ちにロードされることに留意する）。

【００６７】まず、このシナリオにおいて、ユーザ５はマスタおよびスレーブの双方としてデフォルトによって構成されるため、図８における第１のパス８１０を辿る。このパス８１０は、ユーザ５がマスタおよびスレーブの双方である場合（デフォルト）の共同ウェブ閲覧システムの挙動を記述する。このパスの下での制御フロー例として、次に、ユーザ５は、新しいウェブページ（ページＢ）をホストウェブサーバからユーザ５のターゲットブラウザウィンドウに取り込むことができる（８１１）。ユーザ５はマスタとして構成されるため、その他のすべてのスレーブについても、自動的にページＢが各自のターゲットブラウザウィンドウにロードされる（８１２）。次に、たとえば、マスタユーザ３がページＢの底部までスクロールする（８１３）。これに応答して、ユーザ５を含めすべてのスレーブも、ページＢの底部に自動的にスクロールされた各自のターゲットブラウザウィンドウでページＢを見ることになる（８１４）。以下同様（８１５）である。すなわち、ユーザ５は、マスタかつスレーブであるため、すべてのスレーブターゲットブラウザに何が表示されるかを制御することができるとともに、任意の他のマスタが各自のターゲットブラウザに行ったあらゆる変更を見ることができる。

【００６８】ユーザ５が、コントロールパネル２１０におけるスレーブチェックボックス３１３のチェックは外すが、マスタチェックボックス３１２のチェックを残す場合、ユーザ５はここでマスタのみになる（８２０）。この場合、図８における第２のパス８２０を辿る。このパス８２０は、ユーザ５がマスタであるが、スレーブではない場合の共同ウェブ閲覧システムの挙動を記述する。このパスの下での制御フロー例として、すべてのスレーブが現在各自ターゲットブラウザウィンドウにロードされるウェブページＡを有するが、次にマスタユーザ３がここで新しいページであるページＣをユーザ３のターゲットブラウザに取り込む（８２１）ものとする。ユーザ５は、スレーブとして構成されていないためページＡに留まり、ページＣがユーザ５のターゲットブラウザウィンドウにロードされるのを見ることはないが、このセッションにおける他のスレーブは１人残らず、各自のターゲットブラウザウィンドウにページＣがロードされるのを見る（８２２）。次に、ユーザ５がページＡの底部までスクロールするものとする（８２３）。ユーザ５は、マスタとして構成されており、スレーブはあらゆるマスタによる最も近い変更と同期するため、すべてのスレーブは、各自のターゲットブラウザにページＡがロードされて、ページＡの底部までスクロールされるのを突然見る。以下同様（８２５）である。すなわち、ユーザ５は、マスタであるがスレーブではないため、他のマスタが行った変更はいずれも見ないが、すべてのスレーブはユーザ５が行った変更を見ることになる。

【００６９】ユーザ５が、コントロールパネル２１０におけるマスタチェックボックス３１２のチェックは外すが、スレーブチェックボックス３１３のチェックを残す場合、ユーザ５はここでスレーブのみになる（８３０）。この場合、図８における第３のパス８３０を辿る。このパス８３０は、ユーザ５がスレーブであるが、マスタではない場合に、共同ウェブ閲覧システムの挙動を記述する。このパスの下での制御フロー例として、マスタユーザ３が新しいウェブページであるページＤを、ユーザ３のターゲットウェブブラウザ８３１で閲覧するものとする。ユーザ５を含めすべてのスレーブは、各自のターゲットウェブブラウザにページＤがロードされるのを見る（８３２）。次に、ユーザ５がページＥを閲覧するものとする（８３３）。ユーザ５はマスタとして構成されていないため、その他のすべてのスレーブはページＤに引き続き留まる（８３４）。次に、マスタユーザ２がページＦを閲覧するものとする（８３５）。そうすると、ユーザ５を含めすべてのスレーブが、各自のター

ゲットウェブブラウザにページＦが自動的にロードされるのを見る（８３６）。以下同様（８３７）である。すなわち、ユーザ５は、スレーブであるが、マスタではないため、他のマスタが行ったあらゆる変更を見ることになるが、他のスレーブはユーザ５が行ったいずれの変更も見ることはない。

【００７０】ユーザ５は、コントロールパネル２１０においてマスタチェックボックス３１２およびスレーブチェックボックス３１３双方のチェックを外す場合、ここでマスタでもなければスレーブでもなくなる（８４１）。この場合、図８における第４のパス８４０を辿る。このパス８４０は、ユーザ５がマスタでもなくスレーブでもない場合の共同ウェブ閲覧システムの挙動を記述する。この場合、他のユーザによるあらゆる変更はユーザ５によって見られない（８４２）。さらに、新しいウェブページであるページＧの閲覧等のユーザ５によって行われるあらゆる変更は、他のユーザによって見られない（８４３）。したがって、ユーザ５は、現在共同ウェブ閲覧システムに参加していない。ユーザ５は、たとえば、共有するウェブページを見つけたいが、そのページを見つけるためにしばらくネットサーフィンをしなければならない場合に、こうすることを選択しうる。ユーザ５は、共有したいページを見つけるためにネットサーフィンしている間、他のマスタによって行われる変更に邪魔されないように、またユーザ５が共有したいページを探している間にロードする中間ページを他のスレーブが見ないように、マスタおよびスレーブ双方のチェックボックスのチェックを外すことができる。

【００７１】ユーザ５がマスタでもなければスレーブでもない図８の場合を続け、ユーザ５が、共同ウェブ閲覧セッションでスレーブと共有したいページＧを最終的に閲覧する（８４３）。ユーザ５がマスタおよびスレーブの双方として再び参加したい場合、ユーザ５がこの時点においてセッションに再び参加する順序が重要である。ユーザ５が最初にスレーブチェックボックスを再びチェックする場合（８４８）（マスタチェックボックスのチェックが外れたままで）、ユーザ５のターゲットブラウザにおけるページＧは、他のスレーブによって現在閲覧されているウェブページが何のページであれユーザ５のターゲットブラウザにロードされるため、即座に「上書き」される（８４９）。この時点で、ユーザ５は、スレーブとして参加しているが、マスタとしては参加していない（８３０）。これは、ユーザ５がどこか他を閲覧していて、次にセッションで皆が何を閲覧しているのか見たい場合には望み通りの挙動であるが、ユーザ５は、本当にセッションにおける他のすべてのスレーブとページＧを共有したかったものとする。この場合、ユーザ５は、スレーブチェックボックスをチェックする前に、マスタチェックボックス８４４をチェックすべきである（または、ユーザ５は、スレーブチェックボックスをまったくチェックしないよう決めてもよい）。ユーザ５はマスタとしてセッションに参加するため、すべてのスレーブがページＧ（または、ページが何であれ、現在ユーザ５のターゲットウェブブラウザにロードされているページ）を強制的に閲覧させられることをユーザ５に注意する警告が、ユーザ５のウィンドウにポップアップする。この注意警告は、マスタとしてセッションに参加する、または再び参加する誰かが、不用意にかつ意図せずにすべてのスレーブターゲットブラウザを切り換えてしまわないように、予防措置として行われる。ユーザ５が「ＯＫ」をクリックし、すべてのスレーブに、ユーザ５が現在閲覧しているページ（この場合ではページＧ）に切り換えることを本当に意図することを示す場合（８４６）、すべてのスレーブターゲットブラウザに即座にページＧがロードされる（８４７）。この時点で、ユーザ５は、セッションにスレーブではなくマスタとして参加している（８２０）。一方、ユーザ５は、すべてのスレーブを切り換えることを本当に意図するかどうかをポップアップ警告で尋ねられたときに「キャンセル」をクリックすると、マスタでもなくスレーブでもないままである（８４１）。

【００７２】ウェブサーバ上のコントローラとユーザブラウザインスタンスにおけるモニタの間で伝送される情報は、多数のフォーマットであることができる。好ましい実施形態において、かつ明瞭性および簡潔性のため、情報は、そのまま受信者のウェブブラウザで実行可能な動的に生成された実行可能Ｊａｖａ（登録商標）スクリプトステートメントに含まれる。パフォーマンスの向上のためには、より簡潔なフォーマットを使用することができる。相互運用性の向上、および新規および既存のＸＭＬツールおよびプロトコル（ＳＯＡＰ等）の活用のためには、ＸＭＬフォーマットを使用することができる。また、好ましい実施形態では、任意の部分の状態が変化するときは常に、変更された状態の部分だけではなく、動的状態全体がすべてユーザに送信されることにも留意する。これにより、ユーザが、随時共同ウェブ閲覧セッションに参加することができ、最新の変更だけではなくすべての状態を最新にできるように保証する。ここでも、よりよいパフォーマンスのため、別の実施は、交換されるメッセージがより小さいように、変更された状態についての情報を送信するだけであってもよい。この場合、さらなる動的情報を要求し、必要に応じて送信することができる。

【００７３】代替の実施形態では、ユーザのコントロールパネルおよびそれに関連する制御プロトコルを別個のブラウザインスタンスで必ずしも実行する必要がない。ユーザのコントロールパネルおよび制御プログラムはすべて、２つのフレームのうちの一方にロードすることができ、他方のフレームは、共同的に閲覧されているターゲットウェブページを含む。この実施形態の利点は、ブ

ラウザの１つのインスタンスを各ユーザのコンピュータで実行することが必要なだけであることである。フレームアプローチの１つの欠点は、共同的に閲覧可能なウェブページのタイプに制約を課すことである。これは、例として、ウェブページによっては、最上位文書としてではなくフレーム内で閲覧される場合にウェブ作成者の意図するようには挙動しないものがあるためである。２つのブラウザインスタンスを使用することにより、閲覧ページは、まるで共同的に閲覧されていないかのように、それぞれそれ自体のブラウザウィンドウで実行されるため、閲覧ページのコンテンツは任意である。ターゲットウェブページをフレームにロードする他の欠点は、ターゲットウェブページのＵＲＬが、ブラウザの「ロケーション」または「アドレス」フィールドで見ることができず、「お気に入りに追加」または「ブックマークに追加」等のウェブブラウザの機能を使用して、ターゲットウェブページをブックマークすることができないことである。

【００７４】クロスプラットフォームおよびクロスブラウザの独立性を促すため、Ｊａｖａ（登録商標）スクリプトが、ユーザウェブブラウザにおいてこの発明を実施するための記述言語として好ましい。他の任意のプログラミング言語、Ｊａｖａ（登録商標）スクリプトの変形、またはウェブブラウザのドキュメントオブジェクトモデルにアクセス可能な他の機構を、Ｊａｖａ（登録商標）スクリプトの代わりとしてもよい。ドキュメントオブジェクトモデルは、ウェブブラウザの銘柄により異なることがあるため、ブラウザに固有のＪａｖａ（登録商標）スクリプトコードを必要に応じて使用して、ターゲットウェブページ文書の状態の部分にアクセスすることができることに留意する。

【００７５】基本的な共同ウェブ閲覧システムに対する多数のエンハンスメントを実施することができる。所望であれば、標準的な機構を使用して、共同ウェブ閲覧セッションをパスワードで保護するとともに、複数の独立したセッションを維持することができる。共同ウェブ閲覧システムの態様のＧＵＩ表現が動的に閲覧可能であり、管理に使用できるように、管理用のインタフェースをウェブサーバ上で実行されているコントローラに追加することも賢明であり、また簡単明瞭なことである。たとえば、１つのこのようなインタフェースを最初に使用して、新しい専用共同ウェブ閲覧セッションを特定のウェブサイトについてのヘルプを求めているユーザと確立することができる。

【００７６】エンハンスメントの別の例として、送信者に他ユーザのブラウザのさらなる態様を制御させることが可能である。最低限、各送信者は、任意の送信者のウィンドウサイズに対する変更、現在ロードされているウェブページアドレス（ＵＲＬ）、スクロールバー位置、ウェブページが含んでいるあらゆるフレームのＵＲＬおよびスクロールバー位置を反映する。しかし、この機能性は、Ｊａｖａ（登録商標）スクリプトドキュメントオブジェクトモデルがそのブラウザにアクセスを許す他の任意のオブジェクトの追跡に拡張することができる。たとえば、ユーザのテキスト入力を、ウェブページ上のフォームにおけるテキストフィールドに入力されるときに追跡し、それによって任意のユーザがこのようなフィールドに何をタイプしたかをすべてのユーザが見ることが望ましい場合がある。同様に、チェックボックスおよびプルダウンリストボックス等の他の形態の要素は、共同的に変更可能である。任意のテキスト選択も共有することができる。すなわち、マスタが１段落のテキストを選択すると、すべてのスレーブが、各自のブラウザでもその段落のテキストが選択されて見える。

【００７７】好ましい実施形態では、各ユーザがセッションに送信者として、閲覧者として、それとも双方として参加するかを決定することができる。ユーザにこれらすべてのオプションが与えられるわけではない、または１つまたは複数のオプションがパスワードで保護された、この発明の用途があることが予見される。たとえば、オンライン教室の状況では、指導者が唯１人の送信者であり、すべての生徒が閲覧者であることが望ましい。

【００７８】本発明は、コラボレーションセッションにおける共同レベルを大幅に強化する分散ポインタも提供する。分散ポインタは、セッションにおけるすべてのユーザが見ることができ、すべてのユーザが移動させることのできる（任意選択的に）ポインタである。このような分散ポインタなしでは、共同中のユーザが確実に、ウェブページまたはフレームのどの部分について議論されているか知ることが困難でありうる。任意の送信者は、現在共同的に閲覧されているターゲットウェブページ２２０に１つまたは複数の分散ポインタを生成することができる。各ポインタ２２１は、ＨＴＭＬポインタコードをターゲットウェブページに挿入することによって生成される。プリンタコードは、既存のターゲットウェブページの最上部にＨＴＭＬレイヤを生成する。このレイヤは、ユーザが、おそらく特別なキーボードとマウスクリックの組み合わせを使用して示す、ウェブページ上の場所に作成される。

【００７９】各レイヤは、ポインタのイメージを含むが、そうでなければ透明である。次に、送信者は、標準的なマウスのクリック手順とドラッグ手順を用いてポインタをターゲットウェブページの周囲を移動させることができる。各ポインタの存在、可視性、および位置についての状態情報は、他の動的状態（スクロールバー位置等）について上述した方法を使用して、各受信者ブラウザに反映される。任意の送信者が、任意の分散ポインタの可視性および位置を制御することができる。

【００８０】本発明によりコンテンツおよび状態を共有

するインフラストラクチャもまた、当業者には当然明白なはずである機構を使用して、他の機能を促進するように拡張することができる。たとえば、この発明では、コントローラが、これを通して更新が中継されるインテリジェント中央ロケーションとして機能することに留意する。したがって、コントローラは、共同ウェブ閲覧セッションに価値を付加するように強化することができる。エンハンスメントの例としては、純粋にＤＨＴＭＬベースのチャット性能、純粋にＤＨＴＭＬベースの、ユーザにより動的に生成されたウェブコンテンツの共有、およびウェブページの外国ドメインからリモートユーザのブラウザへのロード（ＵＲＬ共有化）がある。たとえば、あるタイプのコンテンツに加入したユーザのみが特定の更新を受信するように、ユーザのカテゴリをサポートする性能をコントローラに追加することは単純明快である。さらに、各更新にタイムスタンプして保存することにより、後に共同閲覧セッションを「再生」することが可能になる。

【００８１】本発明によって提供される枠組みにインテリジェントコントローラを有することで、共同的に閲覧しているページのコンテンツを動的に変換することが可能になる。たとえば、コントローラは、受信者に配布する前に、送信者によって受信される更新を変換するように強化することができる。数例として、コントローラは、異なる参加者によって使用されている異なるブラウザの銘柄またはバージョンについて調整するように、異なる参加者の間での言語の相違を調整するように、またはあるコンテンツへの加入またはセキュリティ方式に基づいて各参加者が利用可能な閲覧を制御するように更新を変換しうる。このような任意の変換の態様をイネーブル、ディセーブル、または制御するために、ボタンまたは（垂れ下がる）プルダウンメニューリスト等の制御手段をモニタコントロールパネルに含めることができる。代替として、専用の管理（おそらくウェブベースの）ＧＵＩインタフェースをコントローラに提供して、このような種類の変換を制御することがより適切な場合もある。

【００８２】好ましい実施形態での連続ポーリング機構をイネーブルするのではなく、おそらくボタンを押下することによって１つの同期を明示的に開始する状況があってもよいことも認識される。これは、単純な「リフレッシュ」（受信者の場合）ボタンおよび／または「更新送信」（送信者の場合）ボタンをモニタコントロールパネルに追加することによって促進することができる。

【００８３】この発明は、軽量技術（ＤＨＴＭＬ）および単純な既存のプロトコル（ＨＴＴＰ、ＨＴＴＰＳ、またはおそらくＳＯＡＰ）を使用して、ファイアウォール侵入問題を回避しながら、コラボレーションを促進することに留意されたい。したがって、この発明は、ハンドヘルド装置等の資源が限られた装置におけるコラボレーションを促進するために有利に用いることができる。このような装置は、たとえば、Ｊａｖａ（登録商標）またはアクティブＸコントローラに依存するより重いアプローチに対しては同程度には適さないかもしれない。より重要なことは、本発明を標準かつ単純な機構を使用することをターゲットとすることにより、潜在的なユーザの基盤ははるかに大きい。

【００８４】本発明は、当業者が本発明を製造し使用できることに関して説明され、特定の適用例および要件のコンテキストにおいて提供された。好ましい実施形態に対する様々な変更形態が、当業者には容易に明らかであり、本明細書に記載の原理は、本発明の精神および特許請求の範囲から逸脱せずに他の実施形態および用途にも適用することができる。したがって、本発明を本明細書に記載の実施形態に限定する意図はなく、本発明は、本明細書に開示した原理および特徴と一致する、併記の特許請求の範囲の最も広い範囲と一致する。

【図面の簡単な説明】

【図１】　本発明による分散型共同ウェブ閲覧システムのブロック図である。

【図２】　モニタおよびターゲットブラウザウィンドウのブロック図である。

【図３】　モニタウェブページ構造のブロック図である。

【図４】　ＨＴＴＰを使用するターゲットモニタ要求のリスト例である。

【図５】　更新ファイルからの擬似コードである。

【図６】　図５の更新ファイルを生成するために、コントローラにより用いられる方法のフロー－チャートである。

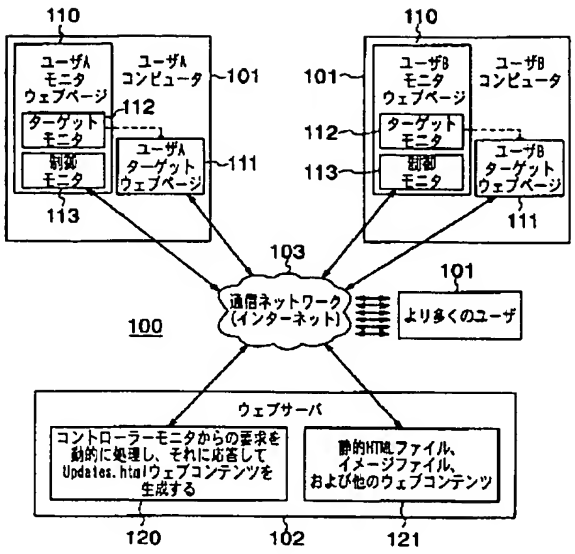
【図７ａ】　コントロールパネルにおいてユーザによる変更を処理し、モニタコントローラおよびモニタのターゲットモニタ構成要素を実施するために、モニタにより用いられる方法のフローチャートである。

【図７ｂ】　コントロールパネルにおいてユーザによる変更を処理し、モニタコントローラおよびモニタのターゲットモニタ構成要素を実施するために、モニタにより用いられる方法のフローチャートある。

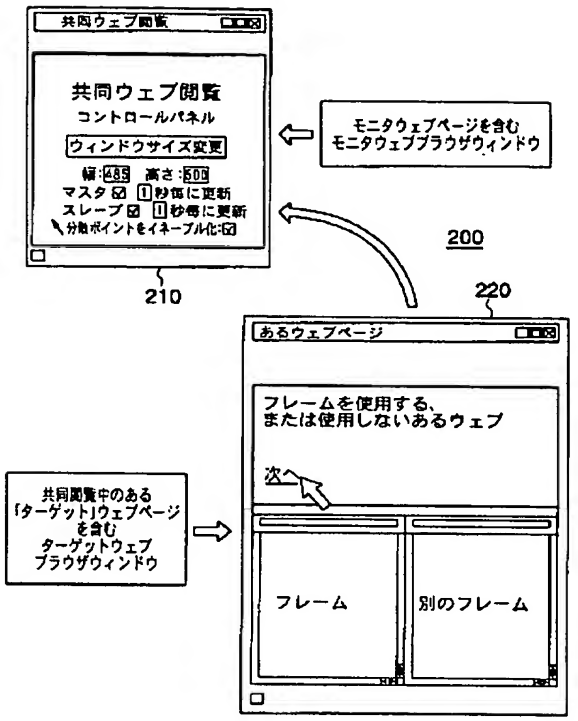
【図７ｃ】　コントロールパネルにおいてユーザによる変更を処理し、モニタコントローラおよびモニタのターゲットモニタ構成要素を実施するために、モニタにより用いられる方法のフローチャートである。

【図８】　ユーザセッション例のフローチャートである。

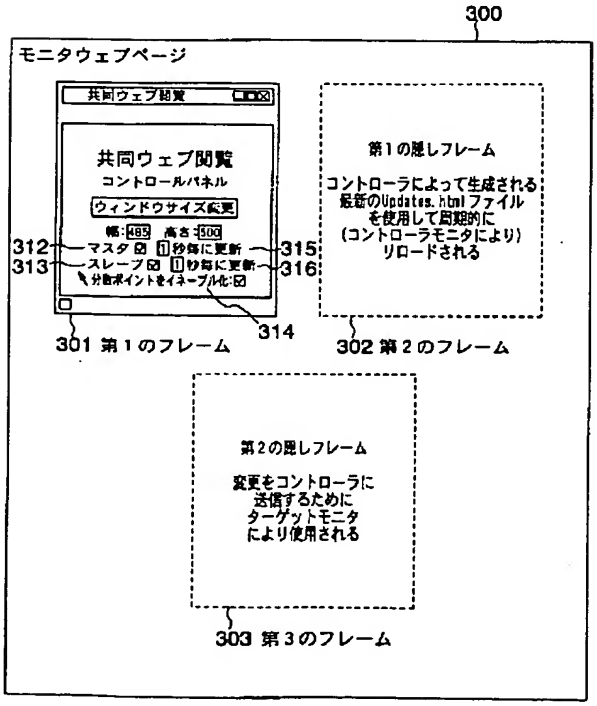
【図１】


110
ユーザA
モニタ
ウェブページ
ユーザA
コンピュータ
101
112
ターゲット
モニタ
制御
モニタ
113
ユーザA
ターゲット
ウェブページ
111
110
101
ユーザB
モニタ
ウェブページ
ユーザB
コンピュータ
112
ターゲット
モニタ
113
制御
モニタ
ユーザB
ターゲット
ウェブページ
111
103
通信ネットワーク
(インターネット)
101
より多くのユーザ
100
ウェブサーバ
コントローラーモニタからの要求を
動的に処理し、それに応答して
Updates.htmlウェブコンテンツを
生成する
静的HTMLファイル、
イメージファイル、
および他のウェブコンテンツ
120
102
121

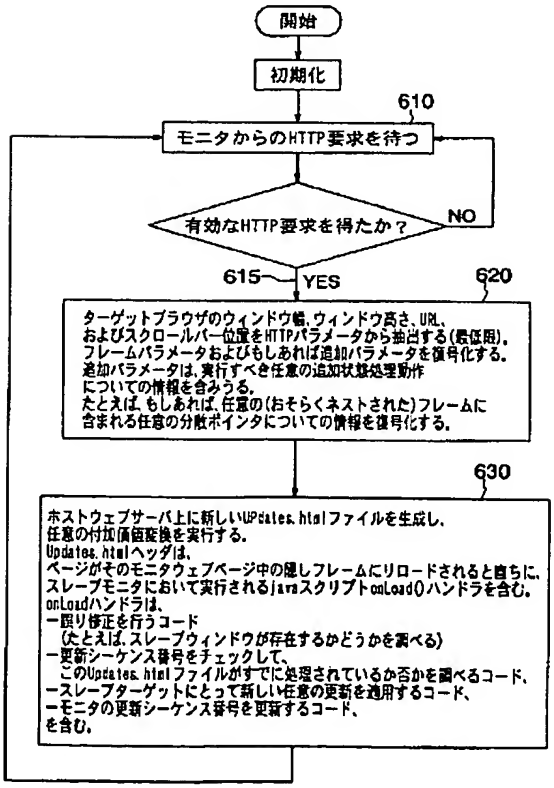
【図２】


共同ウェブ閲覧
共同ウェブ閲覧
コントロールパネル
ウィンドウサイズ変更
幅:485 高さ:500
マスタ 1 秒毎に更新
スレーブ 1 秒毎に更新
分散ポイントをイネーブル化:
210
モニタウェブページを含む
モニタウェブブラウザウィンドウ
200
220
あるウェブページ
フレームを使用する、
または使用しないあるウェブ
次へ
共同閲覧中のある
「ターゲット」ウェブページ
を含む
ターゲットウェブ
ブラウザウィンドウ
フレーム
別のフレーム

【図３】


300
モニタウェブページ
共同ウェブ閲覧
共同ウェブ閲覧
コントロールパネル
ウィンドウサイズ変更
幅:485 高さ:500
312 マスタ 1 秒毎に更新 315
313 スレーブ 1 秒毎に更新 316
分散ポイントをイネーブル化:
314
301 第１のフレーム
第１の隠しフレーム
コントローラによって生成される
最新のUpdates.htmlファイル
を使用して周期的に
(コントローラモニタにより)
リロードされる
302 第２のフレーム
第２の隠しフレーム
変更をコントローラに
送信するために
ターゲットモニタ
により使用される
303 第３のフレーム

【図６】


開始
初期化
610
モニタからのHTTP要求を待つ
有効なHTTP要求を得たか？
NO
615
YES
620
ターゲットブラウザのウィンドウ幅,ウィンドウ高さ,URL,
およびスクロールバー位置をHTTPパラメータから抽出する(最低限),
フレームパラメータおよびもしあれば追加パラメータを復号化する.
追加パラメータは,実行すべき任意の追加状態処理動作
についての情報を含みうる,
たとえば,もしあれば,任意の(おそらくネストされた)フレームに
含まれる任意の分散ポイントについての情報を復号化する.
630
ホストウェブサーバ上に新しいUpdates.htmlファイルを生成し,
任意の付加価値変換を実行する.
Updates.htmlヘッダは,
ページがそのモニタウェブページ中の隠しフレームにリロードされると直ちに,
スレーブモニタにおいて実行されるjavaスクリプトonLoad()ハンドラを含む.
onLoadハンドラは,
－誤り修正を行うコード
(たとえば,スレーブウィンドウが存在するかどうかを調べる)
－更新シーケンス番号をチェックして,
このUpdates.htmlファイルがすでに処理されているか否かを調べるコード,
－スレーブターゲットにとって新しい任意の更新を適用するコード,
－モニタの更新シーケンス番号を更新するコード,
を含む.

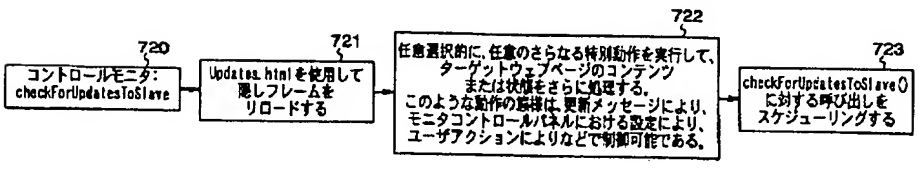
【図４】

401
GET
/servlet/CWControllerServlet?cmd=doUpdates&W=485&H=500&S=http%3A//harm.meitca.com/wwwelevatorcom/&L=0&T=0&P=hnull/null&fn=P.Z[0]-f1&fn=P.Z[1].Z[0]-f2&fn=P.Z[1].Z[1]-f3&fn=P.Z[1]f4&f1.W=0&f1.H=0&f1.S=http%3A//harm.meitca.com/wwwelevatorcom/btn2.htm&f1.L=0&f1.T=0&f1.P=v77/61&f2.W=0&f2.H=0&f2.S=http%3A/harm.meitca.com/wwwelevatorcom/design/layout/head.htm&f2.L=0&f2.T=0&f2.P=v155/15&f3.W=0&f3.H=0&f3.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/gpm3_3.htm&f3.L=287&f3.T=1560&f3.P=v355/1740&f4.W=0&f4.H=0&f4.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/Default.htm&f4.L=0&f4.T=0&f4.P=hnull/1740 HTTP/1.0

402 同じ要求であるが、読みやすさのために改行が挿入されている

GET
/servlet/CWControllerServlet?
cmd=doUpdates
&W=485
&H=500
&S=http%3A//harm.meitca.com/wwwelevatorcom/
&L=0
&T=0
&P=hnull/null
&fn=P.Z[0]-f1
&fn=P.Z[1].Z[0]-f2
&fn=P.Z[1].Z[1]-f3
&fn=P.Z[1]-f4
&f1.W=0
&f1.H=0
&f1.S=http%3A//harm.meitca.com/wwwelevatorcom/btn2.htm
&f1.L=0
&f1.T=0
&f1.P=v77/61
&f2.W=0
&f2.H=0
&f2.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/head.htm
&f2.L=0
&f2.T=0
&f2.P=v155/15
&f3.W=0
&f3.H=0
&f3.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/gpm3_3.htm
&f3.L=287
&f3.T=1560
&f3.P=v355/1740
&f4.W=0
&f4.H=0
&f4.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/Default.htm
&f4.L=0
&f4.T=0
&f4.P=hnull/1740
HTTP/1.0

【図７ｂ】

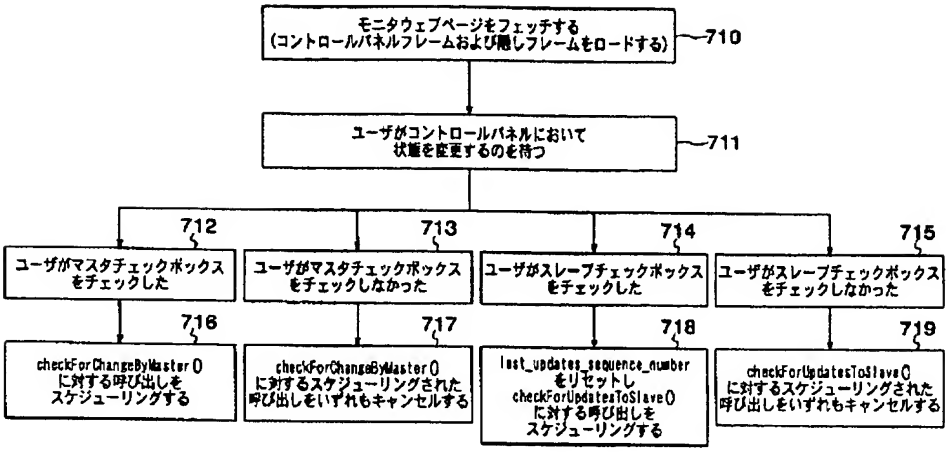


【図５】

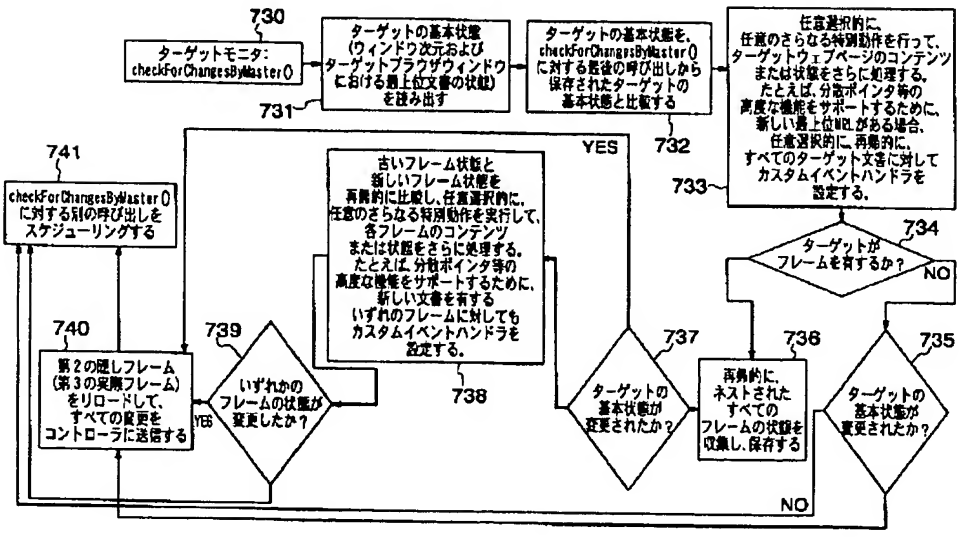
500

```
<HTML><HEAD>
<!-- This file was generated by the Controller on the hostw eb server -->
<TITLE>Collaborative Web Browsing:H idden Updates Frame -- Updates.html</TITLE>
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<SCRIPT LANGUAGE="Javascript">
 var updates_sequence_number = 13;

 var distributedPointerHTML = <<-- HTML code for Distributed Pointer image -->>

 function doOnLoad() {
  if( Cannota ccess Monitor Control Panel web page )
     alert("Cannotf ind Slave Control Panel window");
     return; // Try again after nextu pdates interval
  }

  if( Monitor Control Panel web page has an older commands_sequence_number) {
   // Apply new updates

   //M ake sure the Slave Targetb rowser window is accessible
   if( Slave Target browser window is nota ccessible) {
      alert("Could notf ind Slave window");
      Disable Slave updates in the Monitor Control Panel;
      return;
   }
   if( CurrentS lave Targetw indow dimensions do notm atch new values) {
      Close Slave Targetb rowser window;
      Re-open Slave Targetb rowser window with new width and heightd imensions;
   }
   if( URL olw eb page in Slave Targetw indow does notm atch new URL) {
      Load new URL into Slave Targetb rowser;
      return;/ /waitf or Slave Targetb rowser to load web page atn ew URL
   }
   if( new Slave Targetw indow documentd oes notu se frames) {
      if( Slave Targetw indow's horizontal scrollbar position is old)
         Update Slave Targetw indow's horizontal scrollbar position;
      if( Slave targetw indow's vertical scrollbar position is old)
         Update Slave Targetw indow's vertical scrollbar position;
   } else {
      if( URL of web page in Slave Targetw indow's firstf rame is old) {
         Fetch web page atn ew URL into Slave Targetw indow's firstf rame;
         return;/ /waitf or load
      }
      if( horizontal scrollbar position in Slave Targetw indow's firstf rame is old)
         Update horizontal scrollbar position in Slave Targetw indow's firstf rame;
      if( vertical scrollbar position in Slave Targetw indow's firstf rame is old)
         Update vertical scrollbar position in Slave Targetw indow's first frame;
      if( existence,v isibility or position ofD istributed Pointer in Slave Target
         window's firstf rame is old) {
            Update Distributed Pointer Info for Slave Targetw indow's firstf rame;
      }

      <<-- and so on for sibling frames and any nested frames -->>
   }

   Monitor Control Panel web page's updates_sequence_number = updates_sequence_number;
  }

 }
</SCRIPT>
</HEAD>
<BODY onLoad="doOnLoad()">
<b>Updates.html for updates_sequence_number 13</b>
</BODY></HTML>
```

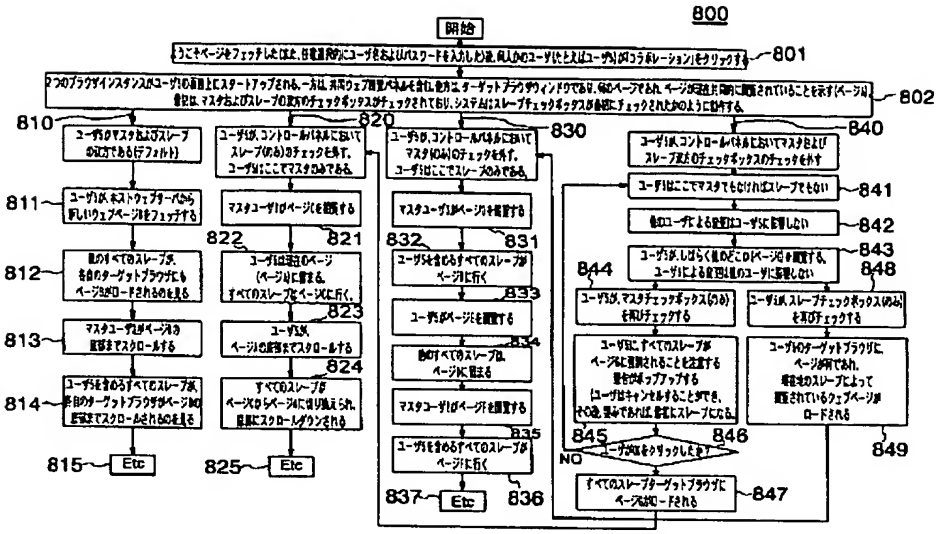
【図７ａ】


モニタウェブページをフェッチする
(コントロールパネルフレームおよび隠しフレームをロードする)
710
ユーザがコントロールパネルにおいて
状態を変更するのを待つ
711
712
ユーザがマスタチェックボックス
をチェックした
713
ユーザがマスタチェックボックス
をチェックしなかった
714
ユーザがスレーブチェックボックス
をチェックした
715
ユーザがスレーブチェックボックス
をチェックしなかった
716
checkForChangeByMaster()
に対する呼び出しを
スケジューリングする
717
checkForChangeByMaster()
に対するスケジューリングされた
呼び出しをいずれもキャンセルする
718
last_updates_sequence_number
をリセットし
checkForUpdatesToSlave()
に対する呼び出しを
スケジューリングする
719
checkForUpdatesToSlave()
に対するスケジューリングされた
呼び出しをいずれもキャンセルする

【図７ｃ】


730
ターゲットモニタ:
checkForChangesByMaster()
ターゲットの基本状態
(ウィンドウ次元および
ターゲットブラウザウィンドウ
における最上位文書の状態)
を読み出す
731
ターゲットの基本状態を、
checkForChangesByMaster()
に対する最後の呼び出しから
保存されたターゲットの
基本状態と比較する
732
任意選択的に、
任意のさらなる特別動作を行って、
ターゲットウェブページのコンテンツ
または状態をさらに処理する。
たとえば、分散ポインタ等の
高度な機能をサポートするために、
新しい最上位NDLがある場合、
任意選択的に、再帰的に、
すべてのターゲット文書に対して
カスタムイベントハンドラを
設定する。
733
734
ターゲットが
フレームを有するか？
NO
YES
741
checkForChangesByMaster()
に対する別の呼び出しを
スケジューリングする
古いフレーム状態と
新しいフレーム状態を
再帰的に比較し、任意選択的に、
任意のさらなる特別動作を実行して、
各フレームのコンテンツ
または状態をさらに処理する。
たとえば、分散ポインタ等の
高度な機能をサポートするために、
新しい文書を有する
いずれのフレームに対しても
カスタムイベントハンドラを
設定する。
738
737
ターゲットの
基本状態が
変更されたか？
736
再帰的に、
ネストされた
すべての
フレームの状態を
収集し、保存する
735
ターゲットの
基本状態が
変更されたか？
740
第２の隠しフレーム
(第３の実際フレーム)
をリロードして、
すべての変更を
コントローラに送信する
739
いずれかの
フレームの状態が
変更したか？
YES
NO

【図８】



フロントページの続き

(72)発明者　アラン・ダブリュ・エセンサー
アメリカ合衆国、マサチューセッツ州、アシュランド、ウィルバー・ドライブ　61

Ｆターム(参考)　5B075 KK02 KK07 NR02 NR13 PQ44
5B085 BE07

【外国語明細書】

1. Title of the Invention

METHOD FOR COLLABORATIVELY BROWSING WEB CONTENT

2. Claims

1. A method for collaboratively browsing web content in a network including a plurality of client computers and a server computer, comprising:

polling static and dynamic states of a web page displayed in a first instance of a web browser;

transmitting the said states to a controller over the said network;

generating, in the said controller, update messages including the said static and dynamic states; and

polling, in a second instance of the web browser, the said controller to receive the said update messages over the said network; and

displaying, in the said second instance of the web browser, the said web page according to the states in the update messages to dynamically synchronize the said second instance of the web browser to the said first instance of the web browser.

2. The method of claim 1 wherein there are multiple instances of the said first and second instances.

3. The method of claim 1 wherein the said polling of the states is performed in a monitor executing in the said first instance of the web browser, and the polling of the controller is performed in the said second instance of the web browser.

4. The method of claim 1 wherein the retrieved target web pages is stored in the server.

5. The method of claim 1 wherein the said polling is performed by polling requests.

6. The method of claim 1 wherein the said first and second instances of the web browser execute in the said client computers, and the said server executes in a server computer.

7. The method of claim 1 wherein the said web page is selected by a user of the said first instance of the web browser.

8. The method of claim 1 wherein the said web page and all instances of the web browser remain unmodified during the collaborative browsing session.

9. The method of claim 4 wherein responses to the said polling requests are loaded into a hidden browser frame, a hidden layer, or another browser window.

10. The method of claim 9 wherein the said hidden browser frame is zero dimensional.

11. The method of claim 1 wherein the said controller transforms dynamically the said update messages.

12. The method of claim 1 further comprising:

polling static and dynamic states of another web page displayed in the said second instance of a web browser;

transmitting the said states to a controller over the said network;

generating, in the said controller, update messages including the said static and dynamic states; and

polling, in the said first instance of the web browser, the said controller to receive the said update messages over the said network; and

displaying, in the said first instance of the web browser, the other web page according to the said states in the said update messages to dynamically synchronize the said first instance of the web browser to the said second instance of the web browser.

3. Detailed Description of Invention

This invention relates generally to the field of computer networks, and in particular to synchronizing access to network content by multiple users.

For some Internet browsing applications, it makes sense to synchronize multiple web browsers so that all users can concurrently view the same "target" web pages. This type of collaborative browsing is typically conducted in conjunction with telephone conversations, class room lectures, or perhaps, with Internet-based voice or text communications among the collaborating users.

A number of collaborative browsing systems are described in the prior art, see for example, U.S. Patent. No. 6,035,332 "METHOD FOR MONITORING USER INTERACTIONS WITH WEB PAGES FROM WEB SERVER USING DATA AND COMMAND LISTS FOR MAINTAINING INFORMATION VISITED AND ISSUED BY PARTICIPANTS;" U.S. Patent No. 5,944,791 "COLLABORATIVE WEB BROWSER;" U.S. Patent No. 6,009,429 "HTML GUIDED WEB TOUR;" U.S. Patent No. 5,991,796 "TECHNIQUE FOR OBTAINING AND EXCHANGING INFORMATION ON THE WORLD WIDE WEB;" U.S. Patent No. 6,151,622 "METHOD AND SYSTEM FOR PORTABLY ENABLING VIEW SYNCHRONIZATION OVER THE WORLD-WIDE WEB USING FRAME HIERARCHIES;" and U.S. Patent No. 5,809,247 "METHOD AND APPARATUS FOR GUIDED TOURING OF INTERNET/INTRANET WEBSITES."

In most prior art collaborative browsing systems, at least one of the users is designated as a "master" or sender, and other users are designated as "slaves" or receivers. Typically, the sender selects the target web pages that are collectively viewed by the viewers. In some systems, the receivers may also have the ability to assume the role of the sender.

As described below, existing collaborative browsing systems may be broadly classified in terms of the mechanisms that are used to direct the viewers to the selected pages.

One class of systems uses a "push" mechanism. There, a web server relies on a special HTTP header (*Content-Type = multipart/mixed; boundary = "someString"*) to keep a dedicated HTTP connection open between the server and each of the browsers executing the receivers' client computers. With this approach, the sender's browser sends the URL of each selected web page to a control program executing on the server. The server's controller then fetches the web page, perhaps processing its content, and pushes the page to each client receiver over the dedicated HTTP connections.

There are a number of problems with this class of systems. The most severe problem is that the push mechanism is not supported by all currently available Internet web browsers, for example, Microsoft's Internet Explorer. Also, the content of the web pages may need to be modified prior to sending in order to direct future requests to the controller. For example, all of the links in the pages may need to be processed so that these requests will be proxied through the controller in the future. This technique is only feasible for simple web pages. Complex pages can have numerous different HTML tags and Javascript commands that could trigger new requests without the control of the sender. It requires a non-trivial amount of effort, and in some cases delay to fully process arbitrarily complex web pages in this manner.

Furthermore, if the receiver directs the browser to some other page, then the dedicated HTTP connection is lost. It is possible to disable all of the links on the target web page by dynamically processing the target web page

before it is sent to the receiver. However, this limits the role of the receiver to that of a totally passive observer. Disabling links also does not work when the receiver uses some other method to load a different web page. More important, modifying pages can alter their intended behavior. This solution also presents difficulties when the target web page contains HTML frames, or when caching is enabled, as described below.

Lastly, with the push mechanism, the receivers cannot have any type of control. The users are merely "pushed" to whatever pages that are selected by the sender.

In another class of collaborative browsing systems, a special monitor is installed on each of the collaborating computers. The monitor opens a dedicated socket connection to the controller. In this case, the sender's monitor tracks the activity of its web browser, and communicates information about updates to the controller where they can be relayed to the receivers' monitors. Apart from being very specific to particular browser implementations, the monitor program needs to be installed on each user computer before the users can collaborate. It is much more desirable to find a solution that does not require the installation of specialized programs, so that users can collaborate immediately even if they are collaboratively browsing for the first time ever.

In another class of collaborative browsing systems, browser "plug-ins" are installed in the web browsers. The plug-in behaves similarly to the monitor described above, and suffers from the same disadvantages.

Another class of collaborative systems, embodied by applications such as Microsoft NetMeeting, facilitates collaboration by essentially sharing all or part of the users' screens across a network. These may be considered complex, platform-specific and "heavy" approaches since they require greater bandwidth, pre-installation of binary executables, pre-registration, and user-training. As with other applications which are based on binary executables that are installed, such non-web-based approaches introduce software installation,

configuration and maintenance issues. Arbitrary users cannot collaborate with such a system unless they all happen to use the same platform and happen to have met the installation, registration and training requirements ahead of time, and have sufficient bandwidth between them. It is desirable to find a solution which does not suffer from any of these requirements.

Another class of systems uses special downloadable programs, such as Java applets or similar technology. The applets execute on each web browser and open a dedicated socket connection to the controller. However, those systems do not work when Java is disabled in the browser. Even if Java is enabled, then it still takes a significantly larger amount of time to load and start the Java environment than it takes to load a simple DHTML web page, degrading performance. Furthermore, this class of systems typically requires embedding code in the target web pages, i.e., the "tag" to load the applets into the target web pages. This limits collaborative browsing only to such specially prepared web pages. If the special code is dynamically embedded into the page when the page is fetched, then there is the risk that modifications to the page will alter its intended behavior. A better solution, which does not require a modification of the target web pages, is desirable.

Existing systems may have other problems when the target web pages use HTML frames. If a new web page is loaded into a frame on the sender's browser, then the top-level URL of the web page is remains the same. In other words, the URL in the browser's "address" or "location" bar does not change. Therefore, a system which only monitors the top-level URL loaded into the sender's browser may not be aware that a frame on the sender's browser is displaying a web page which is different from the web page that is displayed in the corresponding frame on the receivers' browsers. Even if a URL change in a frame of the sender's browser is detected by some means, there is still the problem of forcing the web page at the new URL to be loaded into the corresponding frame for each receiver. It is desirable to find a solution where a

change to any frames on the sender's browser is reflected in the receiver's browser.

For example, U.S. Patent 6,151,622 "Method and system for portably enabling view synchronization over the world-wide web using frame hierarchies," issued to Fraenkel , et al. on November 21, 2000 described static synchronization of client Web browsers to a selected frame by generating a description of a hierarchy of the selected frame. The hierarchy was transmitted over a network environment and duplicated in target frames of the client browsers.

There a number of problems with that type of collaborative browsing. First, the captured frame hierarchy that is disseminated only includes static frame information, specifically the depth and name of the frame and the URL representing the frame content. Nothing is known or captured about the dynamic state of the selected frame. For example, if one of the users scrolls horizontally or vertically, or presses the "Page Down" key, then the views of the participants will no longer be synchronized. It is desirable to provide a collaborative web browsing solution in which the dynamic state of the web pages being viewed in each browser is also synchronized. Then, for example, if any user scrolls or types something into a form field, the browsers of the other users will be updated to reflect these dynamic state changes.

Furthermore the solution provided by Fraenkel et al. requires that the web page to be collaboratively viewed is loaded into a frame. There are a number of disadvantages with that approach. Some web pages will not behave properly if they are loaded into a frame rather than into the top-level document of the browser. Web pages which contain frames themselves are more prone to stop the browser from functioning. Web pages which are forced to be loaded into a frame cannot be book-marked, either. Finally, if a web page is forced to be loaded into a frame, then the URL of that web page will not be visible in the "Address" or "Location" field of the web browser, because only the URL of the top-level web page is shown in this field. It is desirable to provide a

collaborative web browsing solution for which the web pages to be collaboratively viewed can be loaded as the top-level document of the web browser, such that they can be viewed in their "natural environment" just as if they were not being viewed collaboratively.

Finally, the solution provided by Fraenkel et al. puts the intelligence into collaboration scripts which are downloaded into each browser. Another solution would put this intelligence on the server rather than the clients so that the server can add value to the collaborative session. For example, the programs on the server could transform the dynamic or static state of collaboration information before disseminating that information to client browsers. In general, it may be advantageous to keep the intelligence in the server program because this program has ready access to auxiliary sources of information. Such sources of information might include back-end databases, information about the browser brands and versions in use by the clients, the screen resolution of each client, etc. It is desirable to provide a collaborative web browsing solution which is flexible and easily extended to support adding value to the collaborative web browsing sessions.

Prior art systems can have other problems when they rely on a sender's HTTP request for a new target web page to arrive at the host web server. This may be the case when the arrival of an HTTP request at the host is used as the basis to determine when a new target web page request by the sender must be processed. The problem is that web pages may be cached in the host server, or in some intermediate proxy server. If an HTTP request by the sender's browser is satisfied by a cache hit, then the request may not be reflected in the receiver's' web browser.

Therefore, it is desired to provide a collaborative web browsing system that includes dynamic state information and operates for pre-existing and arbitrary target web pages including any number of HTML frames, links, or Javascript. It also desired that the target web pages can be viewed naturally and without modifications that could alter their intended behavior. For

accessibility and ease of use, it is also desired that the system does not require any special binary executables or programs such as Java applets or browser plug-ins.

The present invention provides a flexible mechanism for synchronizing multiple web browsers. The invention enables unmodified web browsers to share unmodified web pages. This collaboration is made possible, in part, because any collaborating user can remotely control significant aspects of the browsers of other users using a "pull" or polling mechanism.

A "sender" browser selects the web pages to be viewed by "receiver" browsers. The receiver browsers can be "controlled" by one or more sender browsers. Any user browser can either be a sender or a receiver, or both. The invention operates by polling dynamic changes in states of the sender's browser, for example, resizing of the sender's browser window, selecting a new target page, scrolling the sender's browser window or frames, tracking pointer positions, and entering text. All changes in the dynamic state are reflected in the browsers of the receivers.

The invention comprises three main components: a controller executing on a host web server, and a monitor and target for each user participating in the collaborative web browsing system, whether as a sender, as a receiver, or as both.

The controller coordinates updates of the dynamic states of the various collaborating browsers. There is no restriction on the language used to implement the controller. For example, the controller can optionally be implemented as a PERL script, or as a Java servlet to provide platform independence.

The monitor includes an HTML web page including HTML frames, a configuration control panel, and Javascript programs. The monitor can execute in an instance of its own browser, or in one of the target's frames. The monitor knows the name of the target browser window, so the Javascript programs can detect any updates to the target window's dynamic state. The sender's monitor

polls the sender's target to detect updates to its dynamic state to be transmitted to the controller. For example, if the sender scrolls the vertical scrollbar down in the target web page, then the sender's monitor detects this scrolling and transmits the scrolling state to the controller. Each receiver's monitor periodically polls the controller to detect the dynamic state updates to be reflected in the receiver's target.

The target is an instance of the browser, or one of its frames, including the target web page that is to be collaboratively viewed. According to the invention, the target can be a regular instance of any standard browser, as such, the target can be used for normal web browsing. Thus, the invention enables pre-existing and arbitrary web pages to be collaboratively viewed by unmodified browsers.

As mentioned above, the sender's monitor, executing in one browser instance or frame, thoroughly examines the state of the target web page that is loaded into the instance of the target browser instance or frame. The monitor performs this polling periodically at configurable time intervals, such as once every second. This examination determines the structure of the currently loaded target web page, and ascertains significant values to be transmitted to the controller. For simple web pages, this information minimally includes the top-level URL loaded into the page, and the positions of the horizontal and vertical scrollbars.

For web pages that use frames, the information also includes the URL used for each frame, and the scrollbar information for each frame. Other state information may include values typed into any forms on the web page, the text that is currently selected, and information related to the target web page. In addition, the dimensions of the target browser windows can be synchronized so that web content appears identically in all of the users' web browsers. Changes to any of these state values are transmitted to the controller in the form of polling requests, for example, HTTP or HTTPS request, or similar notifications in other communication protocols.

Due to security measures which are built into Javascript and standard web browsers, a Javascript program executing in a first browser instance (or frame) may only be allowed to access the detailed information in a second browser instance (or frame) when both instances are loaded with web pages from the same origin. In this case, the web pages, images and programs used to implement the invention should be stored on the same web server that stores the web pages to be collaboratively viewed.

In the case of a single sender, the state of the sender's target is reflected in all of the receivers' targets. In the case of multiple senders, the receiver is synchronized to the state of any of the senders. In particular, the receiver is synchronized to the state of the sender which made an update most recently. Each sender may also behave as a receiver so that all participant browsers are correctly synchronized with each other. A user may elect to be neither a sender nor a receiver. Such might be the case when, for example, that user wishes to leave the collaborative session briefly to browse elsewhere and then to return and re-join the collaborative browsing session at a later time.

When the sender's monitor transmits the update polling request to the controller indicating that some state, such as a scrollbar position, has changed, the controller dynamically generates an update message which contains details about the new state. This updates message is returned, in response to the periodic polling requests, to the receivers' monitors. The state changes are reflected in each receiver's browser instance (or frame), thus insuring that each receiver's browser is synchronized with all other browsers. The frequency at which the monitor polls for changes in state in either the target, in the case of a sender, or the controller, in the case of a receiver, is configurable via the control panel.

The method of the invention uses dynamic HTML techniques and does not require any special programs such as Java applets, browser plug-ins, or other executable programs to be installed on the user computers before collaboration can begin. Therefore, any arbitrary user on the Internet can

collaborate with other users on the Internet immediately by simply visiting an initial web page. Users do not experience any initial delays such as those associated with prior art Java applets. Users may participate even if they are using a device with limited resources which does not support Java and similar "heavyweight" technologies.

The collaborative browsing system according to the invention is platform independent, and the controller executing on the host web server can be written in a platform independent language such as PERL, or Java.

The present invention enables more flexibility than known collaborative web browsing systems. Receivers can periodically browse elsewhere and then rejoin the collaborative session later. Rather than having one sender browser control all interactions, each user can optionally be a sender and therefore enabled to control the web pages to be collaboratively viewed by other receivers. Because the invention is a web-based application, support for new features can be added to the system without requiring any installation, configuration or maintenance of the user computers. Any such additions need only be made to the files on the host web server.

More particularly, the invention provides a method for collaboratively browsing web content in a network including a plurality of client computers and a server computer. Static and dynamic states of a web page displayed in a first sender instance of a web browser are polled. The states are transmitted to a controller executing in a server computer over the network. In response to receiving the update messages, the controller generates update messages including the states. The controller is then polled by a second receiver instance of the web browser to receive the update messages, and the receiver browser can then display, and the web page can be displayed according to the states in the update messages so that the receiver's web browser is dynamically synchronize the sender's web browser.

Figure 1 shows a collaborative web browsing system 100 according to the invention. The system 100 includes one or more client computer (clients)

systems 101, and a server (host) computer system 102 connected to each other by a network 103.

The clients 101 can be personal computers, workstations, laptops, personal digital assistants (PDA), wireless computing device such as cellular telephones and the like executing software programs. Operating system software can be Windows, LINUX, UNIX, NT, and so forth, application software can include Internet applications such as web browsers. The server can execute server software such as the Apache system. The clients and servers typically include input and output devices, and storage sub-systems. The network can be the Internet including the World-Wide-Web (WWW or web), and the links between the clients and server can be wire or wireless. The network can also include intermediate proxy servers and routers.

The three major components of the system 100 include a controller 120 in the server, and a monitor 110 and a target 111 for each of the clients 101. The monitor 110 can include a target monitor 112 and a controller monitor 113. In the case that the client is a multi-user system, the client can include an instance of the monitor and the target for each user.

The controller 120 is an application program executing on the server 102. The server can also store web content 121, such as web pages, image files, video files, etc., to be collaboratively viewed by the users of the system 100. The content can also be stored and cached in the clients, local servers, or the intermediate servers of the network.

The users of the system 100 can dynamically be enabled as "masters" or senders, as "slaves" or receivers, as both, or as neither. The content 121 selected and viewed by the sender is reflected in the targets 111 of all participating receivers. The target monitor 112 is enabled for senders, and the controller monitor 113 is enabled for receivers. The targets are instances of a standard unmodified web browser that can be loaded with various web content 121.

The target monitor 112 uses Javascript programs to periodically poll the dynamic state of the sender's target 111 to ascertain whether any changes in its dynamic state needs to be transmitted as updates to the controller 120. Any sender can load web content 121 to be collaboratively viewed into that sender's target. The controller monitors 113 communicate with the controller 120 using polling requests. The controller monitors fetch update messages generated in response to actions performed by the sender. Note that the present invention does not require that the monitor and target components exist in distinct web browser instances. The invention also allows these components to run in different frames of a single web browser instance, but in the preferred embodiment there are two distinct browser instances.

Figure 2 shows the client side 200 of a preferred implementation of the invention wherein the monitor and target exist in separate browser instances, rather than in separate frames of the same web page. The web browser instance that contains the monitor initially displays a collaborative web browser control panel 210. This control panel is used to configure a collaborative browsing session. The target browser instance 220 is used by the sender to browse web pages that will be collaboratively viewed by receivers.

As shown in Figure 3, the monitor web page 300 comprises three frames 301-303. A first frame 301 is visible. The other two frames 302-303 are "hidden" HTML frames not visible to the user. Effectively, hidden frames are zero-dimensional, i.e., they have no height or width.

The first frame 301 in the monitor web page 300 is the control panel 210. The control panel is used to configure certain aspects of collaborative browsing sessions. The user can use the control panel to specify 311 the dimensions of the target browser window 220. The user can also enable sender features 312, enable receiver features 313, and enable distributed pointers 314, described below. The user can also use the control panel to specify the frequency with which to poll for changes in dynamic state of the target window if the user is enabled as a sender 315, and changes from the controller

if the user is enabled as a receiver 316. The first frame 301 also contains the Javascript programs that implement the target monitor 112 and the controller monitor 113.

The controller monitor periodically polls the controller by making polling requests. In the preferred embodiment, the requests follow the HTTP or HTTPS protocol. The response to this request is a web page that is referred to by the filename "Updates.html." The Updates.html page is loaded into the first hidden frame 302. The content of the Updates.html file indicates whether any changes have been made to the dynamic state of the target web page of any sender users. If some changes have been made, such as when a new web page has been loaded into the target browser window for a sender, then those changes will be applied to the local target window. The Javascript programs in the controller monitor are only enabled for a receiver.

The target monitor 112 periodically polls the dynamic state of the web page that is currently loaded into the target window 220. Any changes to the state of the target web page are transmitted to the controller 120, which will make the changes available for all receivers. In the preferred embodiment, these changes are transmitted to the controller as parameters in HTTP requests. The HTTP response to this request is loaded into the second hidden frame 303. The HTTP response from the controller is used for debugging. The Javascript programs that implement the target monitor are only enabled if the local user is enabled as a sender.

Figure 4 shows an example polling request made by the target monitor 112 when it sends dynamic state updates to the controller 120 in a compact 401 and expanded 402 form. In this example, the HTTP GET method is used. Alternatively, the HTTP POST method may be used, perhaps when there is a large amount of data in the update. The target monitor may dynamically decide whether the GET or POST method is the most appropriate for a particular set of updates.

In the example shown in Figure 4, the HTTP parameter names and values are encoded via simple text substitutions for brevity. For example, strings of the form "frames[" are replaced with strings "Z[". Note that information about each frame in the target window is encoded using query parameter names of the form "fn.d," where "n" is a number that indicates a particular frame and "d" indicates a particular attribute of that frame. For example, "f1.S" indicates the URL of web page that is loaded into the frame designated "f1." The encodings for these frame designations are also included in the query parameters, e.g., "fn=P.Z[1].Z[0]-f2" indicates that attributes for nested frame frames[1].frames[0] will be contained in query parameters with names that start with "f2." It is recognized that other methods for encoding the parameter names and values may be used, and that the method used may be tailored to best match the environment and communication mechanism with which the invention is being used, e.g. if the SOAP protocol is used, the data could be sent using an XML format.

Figure 5 shows an example Updates.html file 500. This is the file that is generated by the controller 120 in response to updates from the target monitor on a sender. This file is periodically fetched by the controller monitor on each receiver. Note that Updates.html is loaded into the second frame 302 of the monitor web page. Note the "onLoad" handler in the pseudo-code. This handler is executed when this web page has finished loading into the browser. It explicitly compares the dynamic state attribute values in the updates with those in the web page in the receiver's target browser. If any attribute value does not match, then it's value is updated, by Javascript code in Updates.html, in the receiver's target browser. Note that in Figure 5 the word "slave" indicates the "receiver".

The flow of control in the Updates.html file 500, which is used to update the receiver's target web page 220, should be evident to one skilled in the art. Sequence numbers are used to verify whether the particular set of changes reflected in this Updates.html file have been processed, yet. Note that

attributes in any sub-frames are recursively updated, too. Browser-specific Javascript commands may be used as needed to update the receiver's target web page 220, because, if necessary, the controller and monitor are able to detect the platform and version information about the web browser's in use using standard Javascript Document Object Model methods and properties. It is recognized that the format and use of Javascript in Figure 5 represents only one of many potential ways of conveying the updates information for transmission to the receivers. This format is used in the preferred embodiment for clarity and simplicity of operation.

Figure 6 shows a method 600 used by the controller 120 to process updates from a sender. The method waits for a polling request 610. When a valid request is received 615, the dynamic state of the target browser is extracted 620 to generate 630 a new Updates.html file 500.

The controller program 120 running on the host web server 102 is designed to wait for valid requests 610 from a monitor. Upon detecting such a request 615, the controller extracts (typically for minimal level of functionality) the target browser's window width, window height, URL, and scrollbar positions from the HTTP parameters. This includes any URL and scrollbar positions for frames, if any, used in the target web page. Additional parameters may also be decoded which may contain information about additional state processing information to be conveyed to the receivers. For example, these additional parameters may contain information about how an HTML form is being filled in, what text is being selected, and the existence and location of any Distributed Pointers, described below, any of which may be available in any possibly nested frame in the target web page.

After extracting the information about updates which a sender has made to the target web page, the controller generates 630 a new Updates.html file 500 on the host web server 102. As shown in Figure 5, this Updates.html file contains a Javascript onLoad event handler which is executed when the Updates.html file has finished loading into the receiver's hidden frame 302.

This onLoad handler includes code to do error checking (e.g. see if the receiver target window is still open and accessible), to check an updates sequence number to see if this particular Updates.html file has been processed, yet, to apply any new updates to the receiver's target web page, and finally to update the receiver monitor's updates sequence number to indicate that this set of updates has been applied.

Figures 7a, 7b, 7c show methods used by the monitor to operate the user's control panel 710-719, methods used to implement the controller monitor 720-723, and methods used to implement the target monitor 730-741.

Figure 7a shows the method used by the monitor to respond to configuration changes made by the user via the control panel. As shown in Figure 7a, initially, the monitor web page, containing the control panel 301 and two hidden frames 302-303, is loaded 710 into the monitor window or frame 210. Standard event handlers associated with form elements in the monitor control panel 210 are used to respond to configuration changes made by the user. This is indicated in Figure 7a at block 711 where the method waits for the user to change some state in the control panel. If the user checks 712 or un-checks 713 the "Master" (sender) checkbox then a call is either scheduled 716 or any existing scheduled calls are canceled 717, respectively, to the TargetMonitor routines in checkForChangesByMaster() 730. Similarly, if the user checks 714 or un-checks 715 the "Slave" (receiver) checkbox then a call is either scheduled 718 or any existing scheduled calls are canceled 719, respectively, to the ControllerMonitor routines in checkForUpdatesToSlave() 720. In addition, if the user checks the Slave checkbox 714, then a "last_updates_sequence_number" counter in the ControllerMonitor is reset 718 so that the receiver will be properly re-synchronized when this checkbox is checked again. Otherwise the ControllerMonitor might think that this receiver is already up-to-date.

Figure 7b shows the method used by the controller monitor 113 in receivers to regularly monitor the controller such that new updates from

senders will be detected and applied. In other words, the checkForUpdatesToSlave() controller monitor routine is used to update the receiver (slave). If the monitor is configured as a receiver (the Slave checkbox 313 is checked), then the checkForUpdatesToSlave() controller monitor routine is regularly called 720.

The frequency with which this routine is called can be configured in the monitor control panel 316. This method simply reloads 721 the first hidden frame 302 in the monitor web page 300 with the most recent Updates.html file 500 which was generated by the controller 120 on the host web server 102. The onLoad handler in the Updates.html file will apply any needed updates, as explained earlier. The controller monitor may optionally perform any additional special operations 722 to further process the content or state of the receiver's target web page. These operations may be used to add additional value to the collaborative web browsing session. Aspects of these operations may be controlled by the updates messages, by settings in the senders or receivers monitor control panel, or by user action, for example. Finally, the controller monitor checkForUpdatesToSlave() routine re-schedules another call to itself 723 after the interval specified in the control panel 316 so that the next set of updates will be applied, too.

Figure 7c shows the method used by the target monitor 112 in senders to regularly monitor the sender's target web page such that any changes will be transmitted to the controller. In other words, the checkForChangesByMaster() target monitor routine is used to check for any changes made by this sender (master) and if so to transmit them to the controller for relaying to receivers. If the monitor is configured as a sender (the Master checkbox 312 is checked) then the checkForChangesByMaster () target monitor routine is regularly called 730.

The frequency with which this routine is called can be configured in the monitor control panel 315. This method first reads the basic (static and dynamic) state of the target window 731, which minimally includes the

window dimensions and state of the top-level document in the target browser window. The state of the top-level document may minimally include the URL loaded into the top-level window, and the positions of the horizontal and vertical scrollbars. Next the target's basic state is compared to the basic state saved in the target monitor during the previous call to checkForChangesByMaster() 732.

Before using the results of that comparison, the target monitor routine may first optionally perform any additional special operations to further process the content or state of the target web page 733. These operations may be used to add additional value to the collaborative web browsing session. Aspects of these operations may be controlled by settings in monitor control panel or by user action, for example. As an example, to support advanced features such as distributed pointers, described below, if there is a new top-level URL detected in the target web window then at this point the target monitor routine could recursively set custom event handler(s) in the target window to detect the existence or movement of such distributed pointers.

Continuing in Figure 7c, the target monitor checkForChangesByMaster () routine checks to see if the target web page uses frames 734. If not, it checks to see whether the target's basic state changed (this was ascertained earlier 732). If the targets basic state did not change and there are no frames, then there are no updates to be transmitted to the controller, so the method simple schedules another call to itself 741 after the interval specified in the control panel 315. If the target does not have frames 735, but it's basic (top-level) state did change, then the method reloads the second hidden frame 303 (the third actual frame) in the monitor web page to transmit all changes to the controller 740. The idea here is that the HTTP request to reload the second hidden frame is directed to the controller and contains the update information as parameters. The controller decodes the update parameters and sends some web page document in response which is loaded into the second hidden frame 303. This

response document is not used apart from transmitting status (e.g. "successfully received updates") and debugging.

Continuing in Figure 7c, if the target monitor does use frames 734, then the checkForChangesByMaster() routine recursively gathers and saves static and dynamic state of all (possibly nested) frames in the target web page 736. If the target's basic state changed 737 (as was ascertained earlier 732), then the second hidden frame is reloaded in order to transmit those changes to the controller 740, as described above. Note that when the target uses frames and the target's basic state has changed, the method does not bother recursively looking for changes in the frames. This is because changes in the basic top-level state of the target must be handled first. As an example, if the top-level URL has changed, then receiver's will have to load the new top-level URL before they can be allowed to change any frame located in the web page specified by that top-level URL.

If the target is using frames, but the target's basic state has not changed since the last time that checkForChangesByMaster() was called 737, then the method recursively compares 738 old and new frame states and optionally performs 738 any additional special operations (as described earlier for the top-level documents 733). Next, continuing the case where the target has frames and the target's basic state has not changed, the method checks to see if any (possibly nested) frame's state has changed 739. If so, the second hidden frame 303 is reloaded in order to transmit those changes to the controller 740, as described earlier.

In any event, the last step of any call to checkForChangesByMaster() 730 is to schedule another call to checkForChangesByMaster() 741 after the interval specified in the control panel 315.

Figure 8 shows an example flow of operations for typical user sessions when the user is operating as a sender and receiver 810, only as a sender 820, only as a receiver 830, and as neither a sender or receiver 840. In this figure, the word "slave" refers to a receiver, and the word "master" refers to a sender.

This figure by no means shows all of the sequences of operations which are covered by the invention, but it conveys enough information to give someone skilled in the art the idea of how it is intended to operate.

As shown in Figure 8, the typical way in which a user might join a collaborative web browsing session. This user will hereby be referred to, arbitrarily, as "user 5" to distinguish him or her from other users who may already be taking part in this collaborative web browsing session. User 5 initially fetches a special "Welcome" web page 801 in which user 5 might optionally enter a username and password before clicking on a "submit" or "Collaborate" button. This invention does not require password-protected pages, but they may be used, if desired, using standard password-protection mechanisms. In the preferred embodiment, the monitor web page and the user's target web page run in distinct web browser instances. Assuming this is the case, then upon clicking the "submit"/"Collaborate" button 801, two new browser instances will open on user 5's screen 802. The first instance contains the monitor control panel 210, and the other contains the target web page 220. For this scenario we will say that initially both the "Master" (sender) checkbox 312 and the "Slave" (receiver) checkbox 313 are checked in the control panel, and the system will behave as if the Slave checkbox was checked first. This implies that at this point user 5's target browser window will be loaded with the web page that is currently being collaboratively viewed. (Note that if the system initially behaved as if the Master checkbox was checked first, then the target browser of every Slave/receiver currently in the session would immediately be loaded with the same page as is initially loaded into user 5's target browser window.)

Since, initially, in this scenario user 5 is configured by default as both a Master and a Slave, the first path 810 in Figure 8 will be traversed. This path 810 describes the behavior of the collaborative web browsing system when user 5 is both a Master and a Slave (which is the default). As an example flow of control under this path, user 5 may then fetch a new web page (page B)

from the host web server into user 5's target browser window 811. Since user 5 is configured as a Master, all of the other slaves will automatically have page B loaded into their target browser windows, too 812. Next, say that master user 3 scrolls to the bottom of page B 813. In response, all slaves, including user 5, will see page B in their target browser windows automatically scrolled to the bottom of page B, too 814. And so on 815 -- since user 5 is a master and a slave, user 5 can both control what appears in all slave target browsers, and can see any changes that any other masters make in their target browsers.

If user 5 un-checks the Slave checkbox 313 in the control panel 210, but leaves the Master checkbox 312 checked, then user 5 will now only be a master 820. In this case, the second path 820 in Figure 8 will be traversed. This path 820 describes the behavior of the collaborative web browsing system when user 5 is a Master, but not a Slave. As an example flow of control under this path, say that all of the slaves currently have web page A loaded into their target browser windows, but then master user 3 now fetches a new page, page C, into user 3's target browser 821. Since user 5 is not configured as a slave, user 5 will stay at page A and will not see page C loaded into user 5's target browser window, although any and all other slaves in this session will 822. Now say that user 5 scrolls down to the bottom of page A 823. Since user 5 is configured as a master, all slaves will suddenly see their target browser's loaded with page A, and then scrolled to the bottom of page A as they become synchronized with the most recent change by any master 824. And so on 825 -- since user 5 is a master but not a slave, user 5 will not see any changes that other master's make, but all slaves will see changes that user 5 makes.

If user 5 un-checks the Master checkbox 312 in the control panel 210, but leaves the Slave checkbox 313 checked, then user 5 will now only be a slave 830. In this case, the third path 830 in Figure 8 will be traversed. This path 830 describes the behavior of the collaborative web browsing system when user 5 is a slave, but not a master. As an example flow of control under

this path, say that master user 3 browses to a new web page, page D, in user 3's target web browser 831. All of the slaves, including user 5, will see their target web browsers loaded with page D 832. Now say that user 5 browses to page E 833. Since user 5 is not configured as a master, all of the other slaves continue to stay at page D 834. Now say that master user 2 browses to page F 835. Then all of the slaves, including user 5 will see their target web browsers automatically loaded with page F 836. And so on 837 -- since user 5 is a slave but not a master, user 5 will see any changes that other masters make, but other slaves will not see any changes that user 5 makes.

If user 5 un-checks both the Master checkbox 312 and the Slave checkbox 313 in the control panel 210, then user 5 will now be neither a master or a slave 841. In this case, the fourth path 840 in Figure 8 will be traversed. This path 840 describes the behavior of the collaborative web browsing system when user 5 is neither a master nor a slave. In this case any changes by other users will not be seen by user 5 842. Furthermore, any changes made by user 5, such as browsing to a new web page, page G, will not be seen by other users 843. Therefore, user 5 is not currently participating in the collaborative web browsing system. User 5 may elect to do this if, for example, they wish to find a web page to share, but they must surf around for awhile to find that page. While user 5 is surfing around to find the page which user 5 wishes to share, user 5 may uncheck both the master and slave checkboxes so that user 5 will not be bothered with changes made by other masters, and so that other slaves will not see intermediate pages that user 5 loads while looking for the page which user 5 wishes to share.

Continuing the case in Figure 8 where user 5 is neither a master nor a slave, say that eventually user 5 browses to page G 843, which user 5 wishes to share with slaves in the collaborative web browsing session. If user 5 wishes to re-join as both a master and a slave, then the order in which user 5 re-joins the session at this point is critical. If user 5 first re-checks the Slave checkbox 848 (leaving the master checkbox un-checked), then page G in user

5's target browser will immediately be "over-written" as user 5's target browser is loaded with whatever web page is currently being viewed by other slaves 849. At this point user 5 is participating as a slave but not a master 830. This may be the desired behavior if user 5 was browsing elsewhere and now wishes to see what everyone else is viewing in the session, but say that user 5 really did want to share page G with all of the other slaves in the session. In that case user 5 should check the master checkbox 844 before checking the slave checkbox (or user 5 may elect not to check the slave checkbox at all). Since user 5 is joining the session as a master, an alert pops up on user 5's window warning user 5 that all slaves will be forced to view page G (or whatever page is currently loaded into user 5's target web browser). This warning alert is done as a precaution so that someone joining or re-joining a session as a master does not accidentally and unintentionally redirect all of the slave target browsers. If user 5 clicks "OK" 846, indicating that user 5 really does intend for all slaves to be re-direct to the page (page G in this case) that user 5 is currently viewing, then all of the slave target browsers are immediately loaded with page G 847. At this point user 5 is participating in the session as a master, but not a slave 820. If, on the other hand, user 5 clicks "Cancel" when asked in the pop-up alert whether user 5 really intends to redirect all of the slaves, then user 5 will remain as neither a master or a slave 841.

The information transmitted between the controller on the web server and the monitor in the user browser instances can be in a number of formats. In the preferred embodiment, and for clarity and simplicity, the information is contained in dynamically generated executable Javascript statements that can be executed in the receiver web browsers as is. To improve performance, a more concise format could be used. To improve interoperability and leverage new and existing XML tools and protocols (such as SOAP), an XML format could be used. Also note that in the preferred embodiment, whenever any part of the state changes, the entire dynamic state is transmitted to all users, not just

the part of the state that changed. This assures that users can join the collaborative web browsing session at any time and get up-to-date with all of the state -- not just the latest changes. Again, for better performance, another implementation can just send information about state that changed so that the messages exchanged would be smaller. In this case, additional dynamic state information can be requested and transmitted as required.

In an alternative embodiment, it is not necessary for the user's control panel, and its associated control programs, to execute in a separate browser instance. The user's control panel and control programs can all be loaded into one of two frames, where the other frame contains the target web page being collaboratively viewed. The advantage of this embodiment is that it only requires one instance of the browser to be running on each user's computer. One disadvantage of the frames approach is that it imposes limitations on the types of web pages that can be collaboratively viewed. This is because, as an example, some web pages do not behave in the way in which the web author intended when they are viewed within a frame rather than as a top-level document. By using two browser instances, the content of the viewed pages is arbitrary because the viewed pages are running in their own browser window, just as if they were not being viewed collaboratively. Other disadvantages of loading the target web page into a frame are that the URL of the target web page will not be visible in the browser's "Location" or "Address" field, and the target web page cannot be bookmarked using a feature of the web browser such as "Add to Favorites" or "Add Bookmark".

To facilitate cross-platform and cross-browser independence, Javascript is preferred as the scripting language to implement this invention in the user web browsers. Any other programming language, variation of Javascript, or other mechanism, which allows access to the Document Object Model of a web browser, may be substituted for Javascript. Note that, since the Document Object Model may differ between web browser brands,

browser-specific Javascript code may be used as needed to access parts of the state of the target web page documents.

A number of enhancements to the basic collaborative web browsing system can be implemented. Where desired, standard mechanisms can be used to password-protect the collaborative web-browsing sessions, and to maintain multiple independent sessions. It is also advisable and straight-forward to add administrative interfaces to the controller running on the web server such that a GUI representation of aspects of the collaborative web browsing system can be dynamically viewed and used for management. For example, one such interface might be used initially to establish a new dedicated collaborative web browsing session with a user seeking help about a particular web site.

As another example of an enhancement, it is possible to allow a sender to control additional aspects of other user's browsers. Minimally, each sender reflects changes to any sender's window size, the currently loaded web page address (URL), the scrollbar positions, and the URL and scrollbar positions of any frames that the web page contains. However, this functionality can be extended to track any other object that the Javascript Document Object Model allows access to for that browser. For example, it may be desirable to track textual user input as it is entered into a text field in a form on a web page, so that all users will see what any user types into such a field. Similarly, other form elements, such as checkboxes and pull-down listboxes can be collaboratively modified. Arbitrary text selections could also be shared. I.e. if a master selects a paragraph of text, all slaves would see that paragraph of text selected in their browsers, too.

The preferred embodiment allows each user to decide whether to participate in a session as a sender, as a viewer, or as both. It is foreseen that there are applications of this invention where users are not given all of these options, or where one or more of the options are password-protected. For example, in an online classroom situation, it may be desired that the instructor is the only sender and that all students are viewers.

The present invention can also provide distributed pointers, which greatly enhance the level of collaboration in the collaborative session. A distributed pointer is a pointer which all users in the session may see and which all users may (optionally) move. Without such distributed pointers, it may be difficult for collaborating users to know for sure which part of a web page or frame is being discussed. Any sender may generate one or more distributed pointers in the target web page currently being collaboratively viewed 220. Each pointer 221 is generated by inserting HTML pointer code into the target web page. The pointer code generates an HTML layer on top of the existing target web page. This layer is created at the place on the web page at which the user indicates, perhaps with a special keyboard and mouse click combination.

Each layer contains an image of a pointer, but is otherwise transparent. The sender can then move the pointers around the target page using standard mouse clicking and dragging procedures. State information about the presence, visibility and position of each pointer is reflected in each of the receiver's browsers using the methods described earlier for other dynamic state (such as scrollbar positions). Any sender can control the visibility and position of any of the distributed pointers.

The infrastructure for sharing content and state according to the present invention can also be extended, using mechanisms that should be reasonably obvious to one skilled in the art, to facilitate other features. For example, note that in this invention, the controller acts as an intelligent central location through which updates are relayed. Therefore the controller may be enhanced to add value to the collaborative web browsing sessions. Example enhancements include a purely DHTML-based chat capability, purely DHTML-based sharing of web content dynamically generated by users, and loading web pages from foreign domains into remote user's browsers (URL sharing). It is straightforward to add capabilities in the controller to support categories of users, such that, for example, only users who have subscribed to

some type of content will receive particular updates. In addition, by time-stamping and saving each update, it becomes possible to "play back" a collaborative browsing session at a later time.

Having an intelligent controller in the framework provided by the present invention allows one to dynamically transform the content of the pages being collaboratively viewed. For example, the controller may be enhanced to transform updates received by a sender before they are disseminated to receivers. As a few examples, the controller might transform updates to adjust for different browser brands or versions being used by different participants, to adjust for language differences between different participants, or to control the views available to each participant based on some content subscription or security scheme. Controls such as buttons or (cascading) pull-down menu lists may be included in the monitor control panel to enable, disable or control aspects of any such transformations. Alternatively, it may be more appropriate to provide a dedicated administrative (presumably web-based) GUI interface to the controller to control these kinds of transformations.

It is also recognized that there may be situations in which a user would rather explicitly initiate one synchronization, perhaps by pressing a button, rather than enabling the continuous polling mechanism in the preferred embodiment. This can be facilitated by adding a simple "Refresh" (for receivers) &/or "Send Updates" (for senders) button to the Monitor Control Panel.

It should be noted that this invention uses lightweight techniques (DHTML) and simple existing protocols (HTTP, HTTPS, or perhaps SOAP) to facilitate collaboration while avoiding firewall penetration problems. Therefore, this invention can advantageously be used to facilitate collaboration on limited-resource devices, such as handheld devices. Such devices might not lend themselves as well to a heavier approach which relies on Java or ActiveX

controls, for example. More importantly, by targeting the invention to use standard and simple mechanisms, the base of potential users is much greater.

The invention is described in terms that enable any person skilled in the art to make and use the invention, and is provided in the context of particular example applications and their requirements. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the principles described herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments described herein, but is to be accorded with the broadest scope of the claims below, consistent with the principles and features disclosed herein.

4. Brief Description of the Drawings

Figure 1 is a block diagram of a distributed collaborative web browsing system according to the invention;

Figure 2 is a block diagram of monitor and target browser windows;

Figure 3 is block diagram of a monitor web page structure;

Figure 4 is an example listing of a target monitor request using HTTP;

Figure 5 is pseudo-code from an update file;

Figure 6 is a flow diagram of a method used by the controller to generate the updates file of Figure 5;

Figures 7a, 7b, 7c are flow diagrams of methods used by the monitor to process changes by the user in the control panel, and to implement the monitor controller and target monitor components of the monitor; and

Figure 8 is a flow diagram of an example user session.

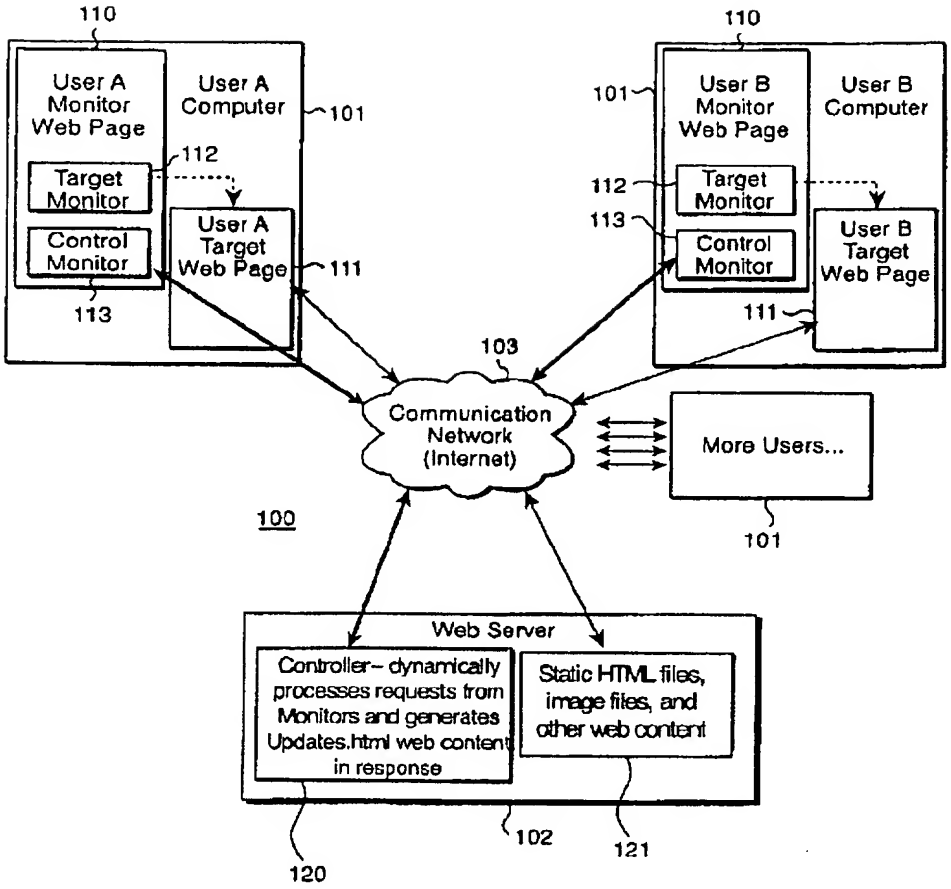

110
User A Monitor Web Page
User A Computer
101
112
Target Monitor
Control Monitor
User A Target Web Page
111
113
110
101
User B Monitor Web Page
User B Computer
112
Target Monitor
113
Control Monitor
User B Target Web Page
111
103
Communication Network (Internet)
More Users...
101
100
Web Server
Controller– dynamically processes requests from Monitors and generates Updates.html web content in response
Static HTML files, image files, and other web content
102
120
121

FIG. 1

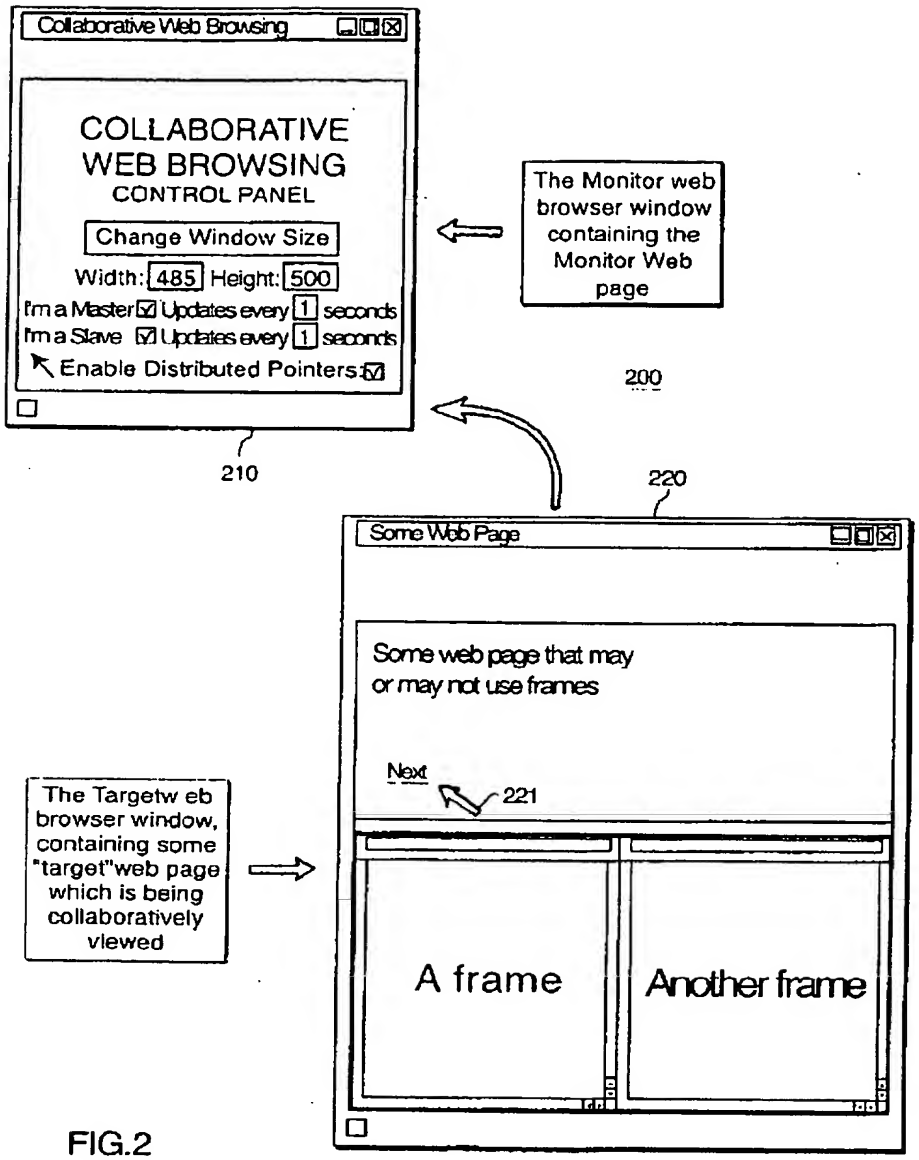

Collaborative Web Browsing
COLLABORATIVE WEB BROWSING
CONTROL PANEL
Change Window Size
Width: 485 Height: 500
I'm a Master ☑ Updates every 1 seconds
I'm a Slave ☑ Updates every 1 seconds
Enable Distributed Pointers ☑
The Monitor web browser window containing the Monitor Web page
200
210
220
Some Web Page
Some web page that may or may not use frames
Next
221
The Targetw eb browser window, containing some "target"web page which is being collaboratively viewed
A frame
Another frame

FIG.2

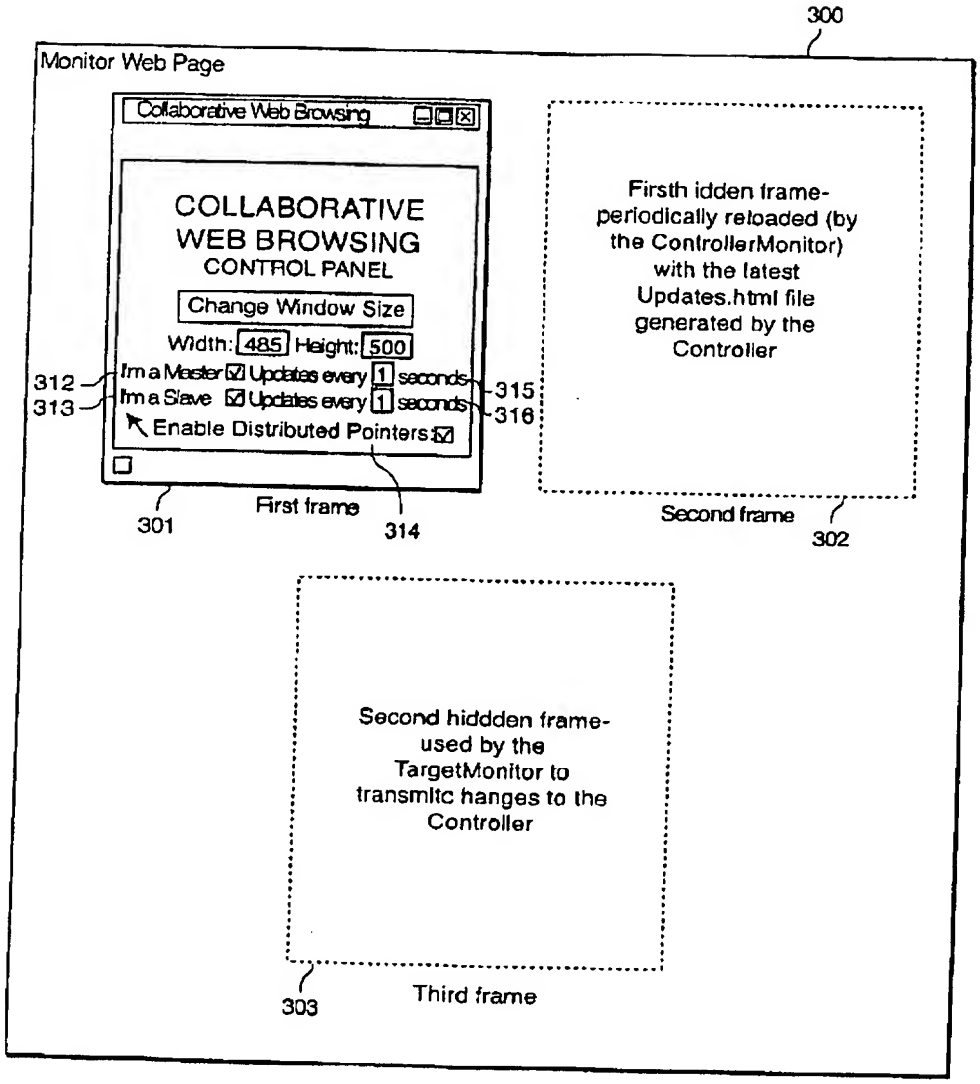

300
Monitor Web Page
Collaborative Web Browsing
COLLABORATIVE
WEB BROWSING
CONTROL PANEL
Change Window Size
Width: 485 Height: 500
I'm a Master ☑ Updates every 1 seconds
I'm a Slave ☑ Updates every 1 seconds
Enable Distributed Pointers ☑
312
313
315
316
301
First frame
314
Firsth idden frame-
periodically reloaded (by
the ControllerMonitor)
with the latest
Updates.html file
generated by the
Controller
Second frame
302
Second hiddden frame-
used by the
TargetMonitor to
transmitc hanges to the
Controller
303
Third frame

FIG.3

401

```
GET
/servlet/CWControllerServlet?cmd=doUpdates&W=485&H=500&S=http%3A//harm.meitca.co
m/wwwelevatorcom/&L=0&T=0&P=hnull/null&fn=P.Z[0]-f1&fn=P.Z[1].Z[0]-f2&fn=P.Z[1].Z[1]-
f3&fn=P.Z[1]f4&f1.W=0&f1.H=0&f1.S=http%3A//harm.meitca.com/wwwelevatorcom/btn2.htm
&f1.L=0&f1.T=0&f1.P=v77/61&f2.W=0&f2.H=0&f2.S=http%3A//harm.meitca.com/wwwelevato
rcom/design/layout/head.htm&f2.L=0&f2.T=0&f2.P=v155/15&f3.W=0&f3.H=0&f3.S=http%3A//
harm.meitca.com/wwwelevatorcom/design/layout/gprn3_3.htm&f3.L=287&f3.T=1560&f3.P=v
355/1740&f4.W=0&f4.H=0&f4.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/
Default.htm&f4.L=0&f4.T=0&f4.P=hnull/1740 HTTP/1.0
```

402 The same request, with line breaks inserted for readbility

```
GET
/servlet/CWControllerServlet?
cmd=doUpdates
&W=485
&H=500
&S=http%3A//harm.meitca.com/wwwelevatorcom/
&L=0
&T=0
&P=hnull/null
&fn=P.Z[0]-f1
&fn=P.Z[1].Z[0]-f2
&fn=P.Z[1].Z[1]-f3
&fn=P.Z[1]-f4
&f1.W=0
&f1.H=0
&f1.S=http%3A//harm.meitca.com/wwwelevatorcom/btn2.htm
&f1.L=0
&f1.T=0
&f1.P=v77/61
&f2.W=0
&f2.H=0
&f2.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/head.htm
&f2.L=0
&f2.T=0
&f2.P=v155/15
&f3.W=0
&f3.H=0
&f3.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/gprn3_3.htm
&f3.L=287
&f3.T=1560
&f3.P=v355/1740
&f4.W=0
&f4.H=0
&f4.S=http%3A//harm.meitca.com/wwwelevatorcom/design/layout/Default.htm
&f4.L=0
&f4.T=0
&f4.P=hnull/1740
HTTP/1.0
```

FIG. 4

500

```
<HTML><HEAD>
<!-- This file was generated by the Controller on the hostw eb server -->
<TITLE>Collaborative Web Browsing:H idden Updates Frame -- Updates.html</TITLE>
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<SCRIPT LANGUAGE="Javascript">
 var updates_sequence_number = 13;

 var distributedPointerHTML = <<-- HTML code for Distributed Pointer image -->>

 function doOnLoad() {
  if( Cannota ccess Monitor Control Panel web page )
     alert("Cannotf ind Slave Control Panel window");
     return; // Try again after nextu pdates interval
  }

  if( Monitor Control Panel web page has an older commands_sequence_number) {
   // Apply new updates

   //M ake sure the Slave Targetb rowser window is accessible
   if( Slave Target browser window is nota ccessible) {
      alert("Could notf ind Slave window");
      Disable Slave updates in the Monitor Control Panel;
      return;
   }
   if( CurrentS lave Targetw indow dimensions do notm atch new values) {
      Close Slave Targetb rowser window;
      Re-open Slave Targetb rowser window with new width and heightd imensions;
   }
   if( URL ofw eb page in Slave Targetw indow does notm atch new URL) {
      Load new URL into Slave Targetb rowser;
      return;/ /waitf or Slave Targetb rowser to load web page atn ew URL
   }
   if( new Slave Targetw indow documentd oes notu se frames) {
      if( Slave Targetw indow's horizontal scrollbar position is old)
        Update Slave Targetw indow's horizontal scrollbar position;
      if( Slave targetw indow's vertical scrollbar position is old)
        Update Slave Targetw indow's vertical scrollbar position;.
   } else {
      if( URL of web page in Slave Targetw indow's firstf rame is old) {
        Fetch web page atn ew URL into Slave Targetw indow's firstf rame;
        return;/ /waitf or load
      }
      if( horizontal scrollbar position in Slave Targetw indow's firstf rame is old)
        Update horizontal scrollbar position in Slave Targetw indow's firstf rame;
      if( vertical scrollbar position in Slave Targetw indow's firstf rame is old)
        Update vertical scrollbar position in Slave Targetw indow's first frame;
      if( existence,v isibility or position ofD istributed Pointer in Slave Target
        window's firstf rame is old) {
          Update Distributed Pointer info for Slave Targetw indow's firstf rame;
      }

      <<-- and so on for sibling frames and any nested frames -->>
   }

   Monitor Control Panel web page's updates_sequence_number = updates_sequence_number;
  }

 }
</SCRIPT>
</HEAD>
<BODY onLoad="doOnLoad()">
<b>Updates.html for updates_sequence_number 13</b>
</BODY></HTML>
```

FIG. 5

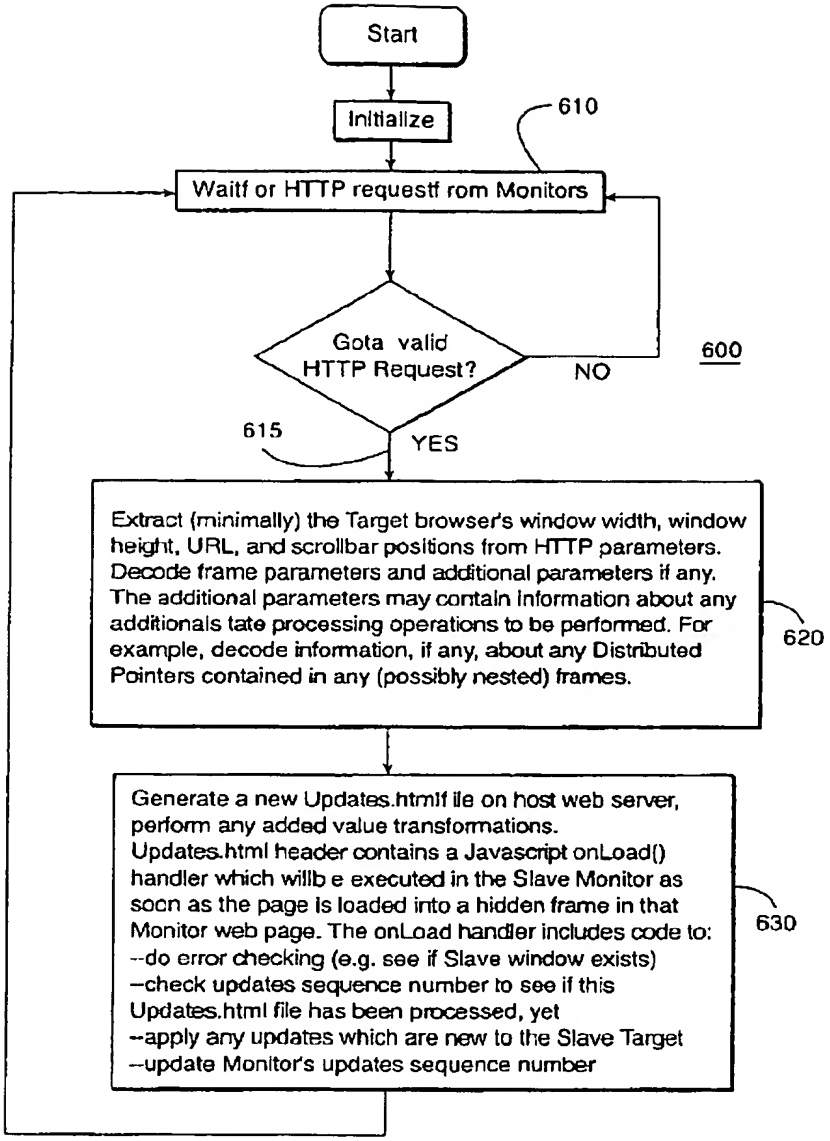

Start
Initialize
610
Waitf or HTTP requestf rom Monitors
Gota valid HTTP Request?
NO
600
615
YES
Extract (minimally) the Target browser's window width, window height, URL, and scrollbar positions from HTTP parameters. Decode frame parameters and additional parameters if any. The additional parameters may contain information about any additionals tate processing operations to be performed. For example, decode information, if any, about any Distributed Pointers contained in any (possibly nested) frames.
620
Generate a new Updates.htmlf ile on host web server, perform any added value transformations. Updates.html header contains a Javascript onLoad() handler which willb e executed in the Slave Monitor as soon as the page is loaded into a hidden frame in that Monitor web page. The onLoad handler includes code to:
--do error checking (e.g. see if Slave window exists)
--check updates sequence number to see if this Updates.html file has been processed, yet
--apply any updates which are new to the Slave Target
--update Monitor's updates sequence number
630

FIG. 6

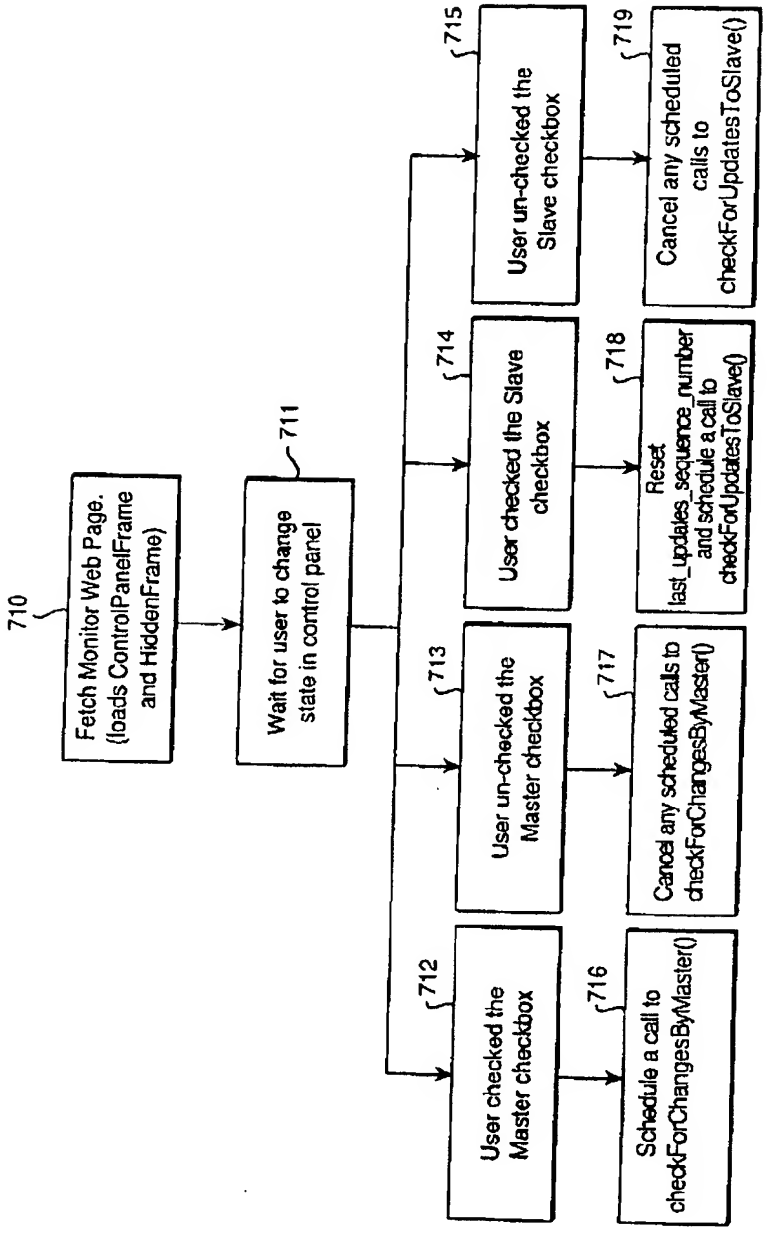

710
Fetch Monitor Web Page. (loads ControlPanelFrame and HiddenFrame)
711
Wait for user to change state in control panel
712
User checked the Master checkbox
713
User un-checked the Master checkbox
714
User checked the Slave checkbox
715
User un-checked the Slave checkbox
716
Schedule a call to checkForChangesByMaster()
717
Cancel any scheduled calls to checkForChangesByMaster()
718
Reset last_updates_sequence_number and schedule a call to checkForUpdatesToSlave()
719
Cancel any scheduled calls to checkForUpdatesToSlave()

FIG. 7a

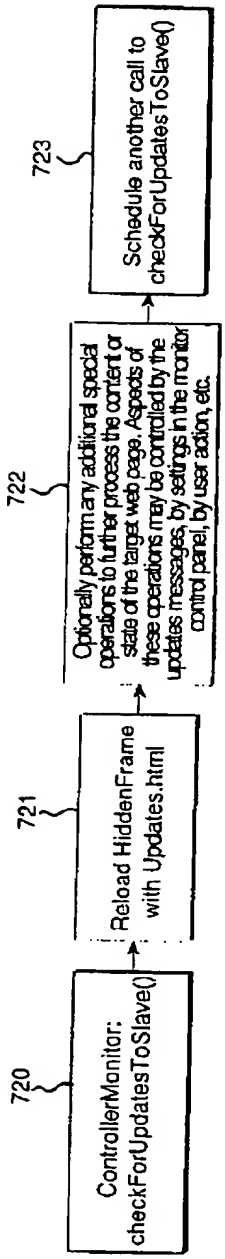

720
ControllerMonitor:
checkForUpdatesToSlave()
721
Reload HiddenFrame
with Updates.html
722
Optionally perform any additional special
operations to further process the content or
state of the target web page. Aspects of
these operations may be controlled by the
updates messages, by settings in the monitor
control panel, by user action, etc.
723
Schedule another call to
checkForUpdatesToSlave()

FIG.7 b

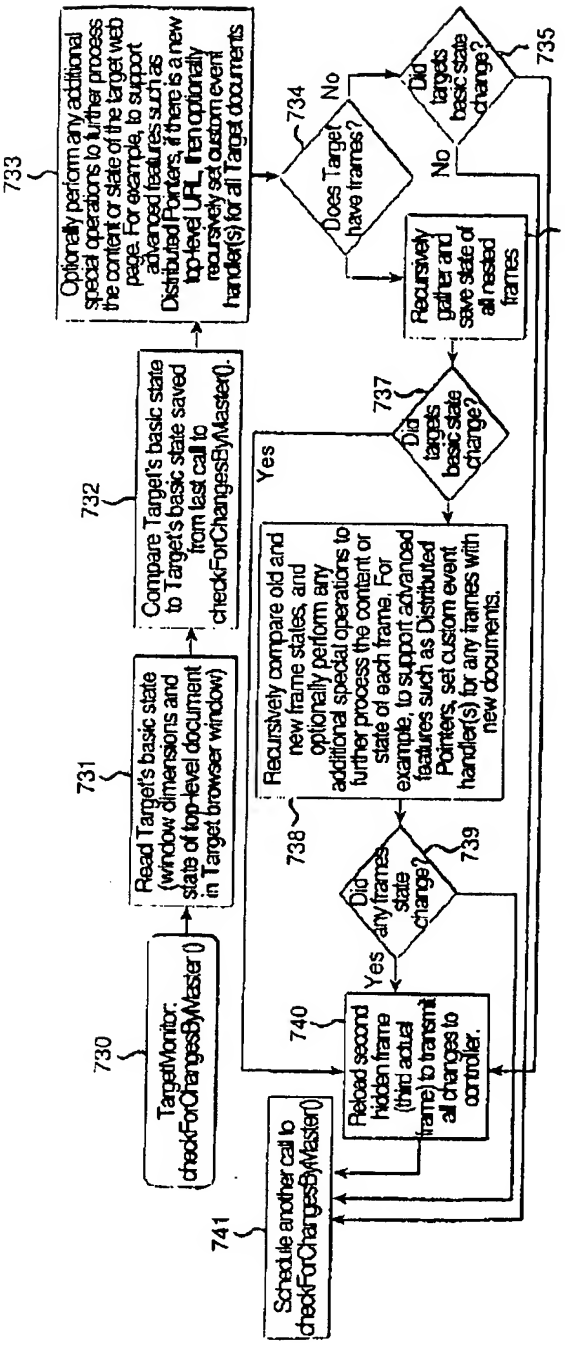

730
TargetMonitor: checkForChangesByMaster()
731
Read Target's basic state (window dimensions and state of top-level document in Target browser window)
732
Compare Target's basic state to Target's basic state saved from last call to checkForChangesByMaster().
733
Optionally perform any additional special operations to further process the content or state of the target web page. For example, to support advanced features such as Distributed Pointers, if there is a new top-level URL, then optionally recursively set custom event handler(s) for all Target documents
734
Does Target have frames?
No
735
Did targets basic state change?
No
736
Recursively gather and save state of all nested frames
737
Did targets basic state change?
Yes
738
Recursively compare old and new frame states, and optionally perform any additional special operations to further process the content or state of each frame. For example, to support advanced features such as Distributed Pointers, set custom event handler(s) for any frames with new documents.
739
Did any frames state change?
Yes
740
Reload second hidden frame (third actual frame) to transmit all changes to controller.
741
Schedule another call to checkForChangesByMaster()

FIG. 7c

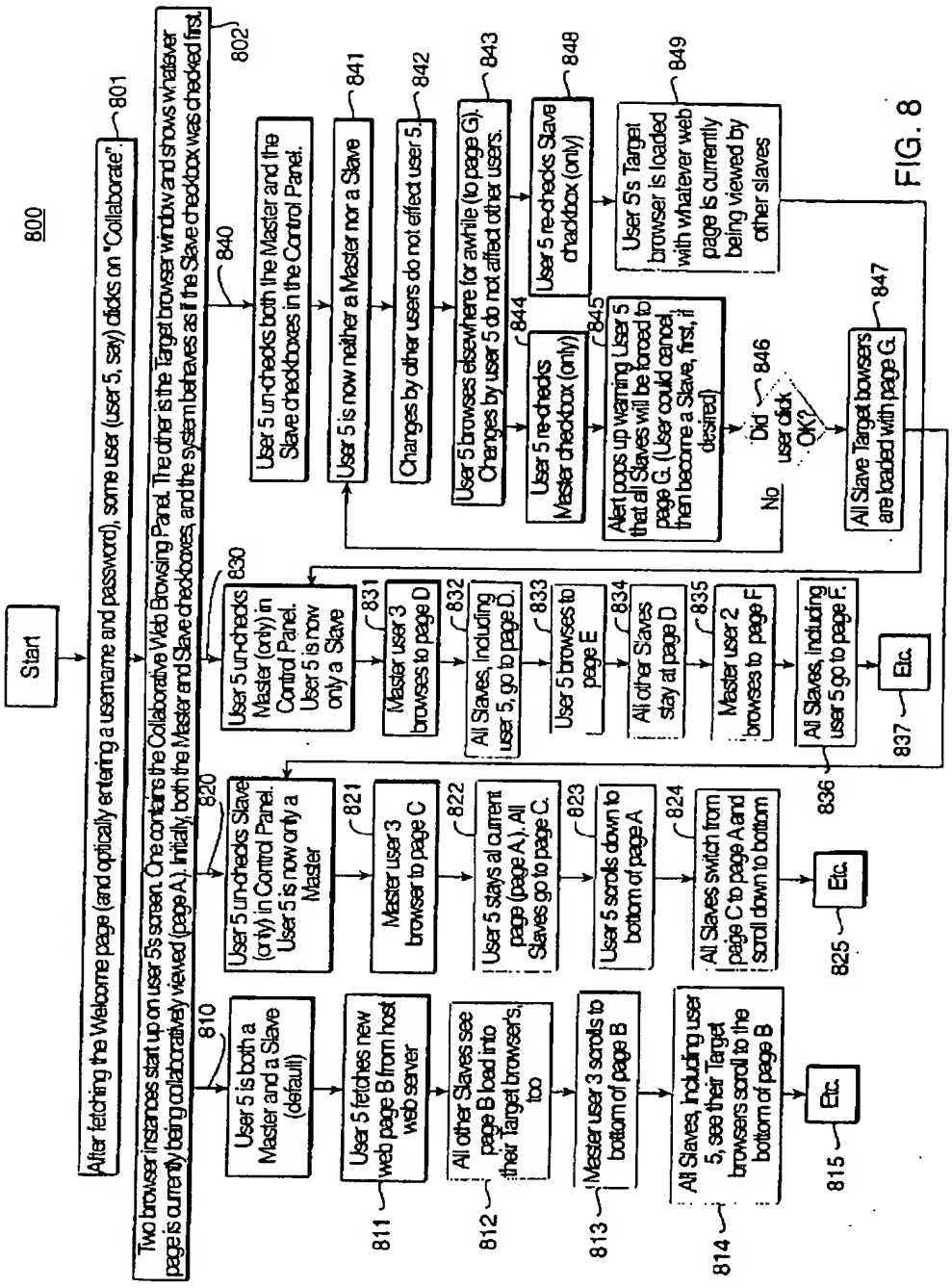

800
Start
After fetching the Welcome page (and optically entering a username and password), some user (user 5, say) clicks on "Collaborate".
801
Two browser instances start up on user 5's screen. One contains the Collaborative Web Browsing Panel. The other is the Target browser window and shows whatever page is currently being collaboratively viewed (page A). Initially, both the Master and Slave checkboxes, and the system behaves as if the Slave checkbox was checked first.
802
User 5 is both a Master and a Slave (default)
810
User 5 fetches new web page B from host web server
811
All other Slaves see page B load into their Target browsers, too
812
Master user 3 scrolls to bottom of page B
813
All Slaves, including user 5, see their Target browsers scroll to the bottom of page B
814
Etc.
815
User 5 un-checks Slave (only) in Control Panel. User 5 is now only a Master
820
Master user 3 browser to page C
821
User 5 stays at current page (page A). All Slaves go to page C.
822
User 5 scrolls down to bottom of page A
823
All Slaves switch from page C to page A and scroll down to bottom
824
Etc.
825
User 5 un-checks Master (only) in Control Panel. User 5 is now only a Slave
830
Master user 3 browses to page D
831
All Slaves, including user 5, go to page D.
832
User 5 browses to page E
833
All other Slaves stay at page D
834
Master user 2 browses to page F
835
All Slaves, including user 5 go to page F.
836
Etc.
837
User 5 un-checks both the Master and the Slave checkboxes in the Control Panel.
840
User 5 is now neither a Master nor a Slave
841
Changes by other users do not effect user 5.
842
User 5 browses elsewhere for awhile (to page G). Changes by user 5 do not affect other users.
843
User 5 re-checks Master checkbox (only)
844
Alert pops up warning User 5 that all Slaves will be forced to page G. (User could cancel, then become a Slave, first, if desired)
845
Did user click OK?
846
No
User 5 re-checks Slave checkbox (only)
848
All Slave Target bowsers are loaded with page G.
847
User 5's Target browser is loaded with whatever web page is currently being viewed by other slaves
849

FIG. 8

1. ABSTRACT

A computerized system enables multiple users of standard Internet web browsers to collaborate by having significant states of their browser, such as which web page is currently being viewed, scrollbar positions, and form values, to be remotely controlled by users of other Internet web browsers. The system uses a monitor to poll the static and dynamic state of the selected pages, and to communicate the state with a controller executing on a web server. The content of the collaboratively viewed pages is arbitrary because viewed pages remain unmodified. Therefore, pre-existing web pages can be collaboratively browsed. Each of the users is optionally a sender or a receiver of selected web pages, and therefore is allowed to control which web pages are collaboratively viewed.

2. Representative Drawing

Fig. 1